

Distributed Cache Service

FAQs

Issue 01
Date 2024-05-20



Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: <https://www.huaweicloud.com/intl/en-us/>

Contents

1 Instance Types/Versions.....	1
1.1 Comparing Redis and Memcached.....	1
1.2 Comparing Versions.....	3
1.3 New Features of DCS for Redis 4.0.....	5
1.4 New Features of DCS for Redis 5.0.....	9
1.5 New Features of DCS for Redis 6.0.....	15
1.6 How Do I View the Version of a DCS Redis Instance?.....	18
1.7 What Are the Differences Between Arm-based and x86-based DCS for Redis?.....	18
2 Feature.....	19
2.1 What Are the CPU Specifications of DCS Instances?.....	19
2.2 What Are Shard and Replica Quantities?.....	20
2.3 Do DCS Redis Instances Limit the Size of a Key or Value?.....	21
2.4 Can I Obtain the Addresses of the Nodes in a Cluster DCS Redis Instance?.....	22
2.5 Does DCS for Redis Support Redis Clusters?.....	22
2.6 What Are Redis Data Eviction Policies?.....	22
2.7 Does DCS for Redis Support Read/Write Splitting?.....	23
2.8 Does DCS for Redis Support Multi-DB?.....	24
2.9 Does DCS Support External Extensions, Plug-ins, or Modules?.....	24
2.10 Does DCS Support Data Persistence? What Is the Impact of Persistence?.....	25
2.11 Do DCS Redis Instances Limite Data Quantity?.....	26
3 Security.....	27
3.1 How Do I Configure a Security Group?.....	27
3.2 Why Can't Security Groups Be Configured for DCS Redis 4.0/5.0/6.0 Basic Edition Instances?.....	29
3.3 How Can I Secure My DCS Redis Instances?.....	30
3.4 Does DCS for Redis Support SSL Encrypted Transmission?.....	31
3.5 How Do I Enable or Disable SSL for Public Access to a DCS Redis 3.0 Instance?.....	31
3.6 Does DCS Support Cross-AZ Deployment?.....	32
3.7 Is a Password Required for Accessing an Instance? How Do I Set a Password?.....	32
3.8 What Is Sentinel?.....	33
3.9 Does DCS Manage Instances in Sentinel Mode?.....	33
4 Client and Network Connection.....	35
4.1 Does DCS Support Public Access?.....	35

4.2 Troubleshooting Redis Connection Failures.....	35
4.3 Does DCS Support Cross-VPC Access?.....	39
4.4 Will I Be Charged for the EIP Used for Public Access to a DCS Redis Instance?.....	39
4.5 Why Is "(error) NOAUTH Authentication required" Displayed When I Access a DCS Redis Instance?..	39
4.6 What Should I Do If Access to DCS Fails After Server Disconnects?.....	39
4.7 Why Do Requests Sometimes Time Out in Clients?.....	40
4.8 What Should I Do If an Error Is Returned When I Use the Jedis Connection Pool?.....	40
4.9 How Do I Access a DCS Redis Instance Through Redis Desktop Manager?.....	42
4.10 What If "ERR Unsupported CONFIG subcommand" is Displayed in SpringCloud?.....	43
4.11 What Can I Do If I Fail to Access a DCS Instance Using Its Domain Name Address?.....	44
4.12 Can I Access DCS Instances in a Local Environment?.....	44
4.13 What Should Be Noted When Using Redis for Pub/Sub?.....	45
4.14 Why Is Public Access of My DCS Redis Instance Unintentionally Disabled?.....	45
4.15 What Can I Do If Error "Cannot assign requested address" Is Returned When I Access Redis Using connect?.....	45
4.16 Connection Pool Selection and Recommended Jedis Parameter Settings.....	46
4.17 What Can I Do If a Lettuce 6.x Client Is Incompatible with My DCS Instance?.....	51
4.18 Should I Use a Domain Name or an IP Address to Connect to a DCS Redis Instance?.....	52
4.19 Is the Read-only Address of a Master/Standby Instance Connected to the Master or Standby Node?	53
5 Redis Usage.....	54
5.1 Can I Change the CPU Architecture?.....	54
5.2 Can I Change the AZ for an Instance?.....	56
5.3 Can I Change the VPC and Subnet for a DCS Redis Instance?.....	59
5.4 Can I Customize or Change the Port for Accessing a DCS Instance?.....	59
5.5 Can I Modify the Connection Addresses for Accessing a DCS Instance?.....	60
5.6 Why Do I Fail to Delete an Instance?.....	60
5.7 Why Does It Take a Long Time to Start a Cluster DCS Instance?.....	60
5.8 What Should I Do If an Error Occurs in redis_exporter?.....	61
5.9 What Is Reserved Memory? How Do I Configure Reserved Memory?.....	61
5.10 Why Is Available Memory Smaller Than Instance Cache Size?.....	62
5.11 Why Is Redisson Distributed Lock Not Supported by DCS Proxy Cluster Redis 3.0 Instances?.....	62
5.12 Does DCS for Redis Provide Backend Management Software?.....	62
5.13 Can I Recover Deleted Data of a DCS Instance?.....	63
5.14 Why Is Available Memory of Unused DCS Instances Less Than Total Memory and Why Is Memory Usage of Unused DCS Instances Greater Than Zero?.....	63
5.15 How Do I Check Redis Memory Usage?.....	63
5.16 Why Is the Capacity or Performance of a Shard of a Redis Cluster Instance Overloaded When That of the Instance Is Still Below the Bottleneck?.....	66
5.17 Why Does an OOM Error Occur During a Redis Connection?.....	66
5.18 What Clients Can I Use for Redis Cluster in Different Programming Languages?.....	66
5.19 Why Do I Need to Configure Timeout for Redis Cluster?.....	68
5.20 Why Am I Seeing a Timeout Error When Reading Data from Redis?.....	69

5.21 Explaining and Using Hash Tags.....	70
5.22 Why Does a Key Disappear in Redis?.....	71
5.23 Will Cached Data Be Retained After an Instance Is Restarted?.....	71
5.24 How Do I Know Whether an Instance Is Single-DB or Multi-DB?.....	71
5.25 What Are the Constraints on Implementing Multi-DB on a Proxy Cluster Instance?.....	72
5.26 How Do I Create a Multi-DB Proxy Cluster Instance?.....	73
6 Instance Scaling and Upgrade.....	75
6.1 Can I Upgrade Version for a DCS Redis Instance, for Example, from Redis 4.0 to Redis 5.0?.....	75
6.2 How Do I Upgrade a Single-Node or Master/Standby Redis 3.0 Instance?.....	75
6.3 Are Services Interrupted If Maintenance is Performed During the Maintenance Time Window?.....	79
6.4 Are Instances Stopped or Restarted During Specification Modification?.....	79
6.5 What DCS Instance Type Changes Are Supported?.....	79
6.6 Are Services Interrupted During Specification Modification?.....	80
6.7 Why Do I Fail to Modify the Specifications for a DCS Instance?.....	86
6.8 How Do I Reduce the Capacity of a DCS Instance?.....	87
6.9 How Do I Add Shards to a Cluster DCS Redis Instance Without Changing the Memory?.....	88
6.10 How Do I Handle an Error When I Use Lettuce to Connect to a Redis Cluster Instance After Specification Modification?.....	89
6.11 Can I Expand a Single Shard of a Cluster Instance (Scale-Up)?.....	91
7 Data Backup, Export, and Migration.....	93
7.1 How Do I Export DCS Redis Instance Data?.....	93
7.2 Why Is Memory of a DCS Redis Instance Unchanged After Data Migration Using Rump, Even If No Error Message Is Returned?.....	93
7.3 Can I Export Backup Data of DCS Redis Instances to RDB Files on the Console?.....	94
7.4 Why Are Processes Frequently Killed During Data Migration?.....	94
7.5 Where Are DCS Instance Backup Files Stored? How Are They Charged?.....	94
7.6 Is All Data in a DCS Redis Instance Migrated During Online Migration?.....	94
7.7 When Will AOF Rewrites Be Triggered?.....	94
7.8 What Are the Common Causes of Redis Migration Failures?.....	95
7.9 Can I Migrate Data to Multiple Target Instances in One Migration Task?.....	95
7.10 How Do I Enable the SYNC and PSYNC Commands?.....	96
7.11 Why Does Migration Task Creation Fail?.....	96
7.12 Will the Same Keys Be Overwritten During Data Migration or Backup Import?.....	96
8 Big/Hot Key Analysis and Expired Key Scan.....	97
8.1 What Are Big Keys and Hot Keys?.....	97
8.2 What Is the Impact of Big Keys or Hot Keys?.....	98
8.3 How Do I Avoid Big Keys and Hot Keys?.....	99
8.4 How Do I Analyze the Hot Keys of a DCS Redis 3.0 Instance?.....	101
8.5 How Do I Detect Big Keys and Hot Keys in Advance?.....	101
8.6 How Does DCS Delete Expired Keys?.....	103
8.7 How Long Are Keys Stored? How Do I Set Key Expiration?.....	103
8.8 Why Does Memory Usage Decrease After Big Key Analysis Is Performed on Redis?.....	104

9 Redis Commands.....	105
9.1 Does DCS for Redis Support Command Audits?.....	105
9.2 How Do I Clear Redis Data?.....	105
9.3 How Do I Find Specified Keys and Traverse All Keys?.....	106
9.4 Why Do I Fail to Execute Some Redis Commands?.....	106
9.5 Why is "permission denied" Returned When I Run the Keys Command in Web CLI?.....	107
9.6 How Do I Rename High-Risk Commands?.....	107
9.7 Does DCS for Redis Support Pipelining?.....	107
9.8 Does DCS for Redis Support the INCR and EXPIRE Commands?.....	108
9.9 Why Does a Redis Command Fail to Take Effect?.....	108
9.10 Is There a Time Limit on Executing Redis Commands? What Will Happen If a Command Times Out?	108
9.11 Can I Configure Redis Keys to Be Case-Insensitive?.....	109
9.12 Can I View the Most Frequently Used Redis Commands?.....	109
9.13 Common Web CLI Errors.....	109
10 Monitoring and Alarm.....	110
10.1 Why Is CPU Usage of a DCS Redis Instance 100%?.....	110
10.2 How Do I View Current Concurrent Connections and Maximum Connections of a DCS Redis Instance?.....	112
10.3 What Should I Do If the Monitoring Data of a DCS Redis Instance Is Abnormal?.....	112
10.4 Why Is Used Memory Greater Than Available Memory?.....	113
10.5 Why Does Bandwidth Usage Exceed 100%?.....	113
10.6 Why Is the Rejected Connections Metric Displayed?.....	114
10.7 Why Is Flow Control Triggered? How Do I Handle It?.....	114
11 Master/Standby Switchover.....	116
11.1 When Does a Master/Standby Switchover Occur?.....	116
11.2 How Does Master/Standby Switchover Affect Services?.....	116
11.3 Does the Client Need to Switch the Connection Address After a Master/Standby Switchover?.....	117
11.4 How Does Redis Master/Standby Replication Work?.....	117
12 Instance Creation and Permissions.....	118
12.1 Why Do I Fail to Create a DCS Redis or Memcached Instance?.....	118
12.2 Why Can't I View the Subnet and Security Group Information When Creating a DCS Instance?.....	118
12.3 Why Can't I Select the Required Enterprise Project When Creating a DCS Instance?.....	118
12.4 Why Can't an IAM User See a New DCS Redis Instance?.....	119
13 Memcached Usage.....	121
13.1 Can I Dump DCS Memcached Instance Data for Analysis?.....	121
13.2 What Memcached Version Is Compatible with DCS for Memcached?.....	121
13.3 What Data Structures Does DCS for Memcached Support?.....	121
13.4 Does DCS for Memcached Support Public Access?.....	121
13.5 Can I Modify Configuration Parameters of DCS Memcached Instances?.....	122
13.6 What Are the Differences Between DCS for Memcached and Self-Hosted Memcached?.....	122

13.7 What Policies Does DCS for Memcached Use to Deal with Expired Data?.....	122
13.8 How Should I Select AZs When Creating a DCS Memcached Instance?.....	123

1 Instance Types/Versions

1.1 Comparing Redis and Memcached

Redis and Memcached are both popular open-source in-memory databases which are easy to use and provide higher performance than relational databases.

How can I select between the two key-value databases?

Memcached is suitable for storing simple data structures, whereas Redis is suitable for storing more complex, larger data that requires persistency.

For details, see the following table.

Table 1-1 Differences between Redis and Memcached

Item	Redis	Memcached
Latency	In-memory database with sub-millisecond latency	In-memory database with sub-millisecond latency
Ease of use	Simple syntax and easy to use	Simple syntax and easy to use
Distributed storage	Horizontal expansion in cluster mode	Supported
Multi-language client	Supports client connections in more than 30 languages including Java, C, and Python.	Supports client connections in more than 10 languages including Java, C, and Python.
Thread/Process	Single-core and single-thread Single-thread communication, avoiding unnecessary context switching and contention Non-blocking I/O (I/O multiplexing) is used to reduce resource consumption when multiple clients are connected.	Multi-thread and scalable The Memcached performance can be improved by increasing the number of CPUs. There is an obvious performance advantage in the scenario where the value of key is great.

Item	Redis	Memcached
Persistent storage	Supported Each write operation (adding, deleting, or modifying data) can be recorded on disk (AOF file).	Supported NOTE Persistence is not supported by open-source Memcached, but is supported by HUAWEI CLOUD DCS for Memcached.
Data structure	Supports complex data structures such as hash, list, set, and sorted set, catering to various scenarios.	Supports simple strings.
Lua script support	Supported	Not supported
Snapshot backup	Supported Snapshots are generated periodically. Therefore, there is no guarantee that data will not be lost. Redis forks a subprocess to generate snapshots. When there is a large amount of data, the Redis service may be interrupted for a short time.	Not supported
Data migration	Supported Data can be backed up and migrated to a new Redis instance through RDB snapshot restoration or AOF file playback.	Not supported
Key value restriction	The value of a key can be up to 1 GB.	1 MB
Multiple databases	A single-node or master/standby DCS Redis instance supports up to 256 Redis databases. A Proxy Cluster or Redis Cluster instance supports only one database, that is, DB0.	Not supported.

Based on the preceding comparison, both the Redis and Memcached are easy to use and have high performance. However, Redis and Memcached are different in data structure storage, persistence, backup, migration, and script support. You are advised to select the most appropriate cache engine based on actual application scenarios.

 **NOTE**

Memcached is suitable for caching scenarios of small amount of static data, where data is only read without further computing and processing, for example, HTML code snippets.
Redis has richer data structures and wider application scenarios.

1.2 Comparing Versions

When creating a DCS Redis instance, you can select the cache engine version and the instance type.

 **NOTE**

DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 4.0, 5.0, or 6.0 instead.

- **Version**

DCS supports Redis 6.0, 5.0, 4.0, and 3.0. [Table 1-2](#) describes the differences between these versions. For details about the new features of Redis 4.0 and 5.0, see [New Features of DCS for Redis 4.0](#) and [New Features of DCS for Redis 5.0](#).

Table 1-2 Differences between Redis versions

Feature	Redis 3.0	Redis 4.0 & Redis 5.0	Redis 6.0
Open-source compatibility	Redis 3.0.7	Redis 4.0.14 and 5.0.14, respectively NOTE DCS Redis 5.0 instances created before February 2022 are compatible with open-source Redis 5.0.9. <ul style="list-style-type: none"> • For details about how to query the open-source version, see How Do I View the Version of a DCS Redis Instance? • To use Redis 5.0.14, create another instance. Currently, the Redis version cannot be upgraded. 	Basic edition: Redis 6.2.7 Professional edition: KeyDB 6.0.16
Instance deployment mode	Based on VMs	Containerized based on physical servers	Containerized based on physical servers
CPU architecture	x86	x86	x86

Feature	Redis 3.0	Redis 4.0 & Redis 5.0	Redis 6.0
Time required for creating an instance	3–15 minutes, or 10–30 minutes for cluster instances.	8 seconds	8 seconds
QPS	100,000 QPS per node	100,000 QPS per node	Basic edition: 150,000 QPS per node Professional edition: 400,000 QPS per node
Public network access	Supported	Not supported	Not supported
Domain name connection	Supported in VPC	Supported in VPC	Supported in VPC
Visualized data management	Not supported	Web CLI for Redis access and data management.	Web CLI for Redis access and data management.
Instance type	Single-node, master/standby, and Proxy Cluster	Single-node, master/standby, Proxy Cluster, and Redis Cluster	Basic edition: single-node, master/standby, and Redis Cluster Professional edition: master/standby
Instance total memory	Ranges from 2 GB to 1024 GB.	Regular specifications range from 2 GB to 1024 GB. Small specifications of 128 MB, 256 MB, 512 MB, and 1 GB are also available for single-node and master/standby instances.	4 GB, 8 GB, 16 GB, 32 GB, and 64 GB (128 MB, 256 MB, 512 MB, and 1 GB are additionally supported for single-node and master/standby instances)

Feature	Redis 3.0	Redis 4.0 & Redis 5.0	Redis 6.0
Capacity expansion/reduction	Online capacity expansion and reduction	Online capacity expansion and reduction	Online capacity expansion and reduction
Backup and restoration	Supported for master/standby and Proxy Cluster instances	Supported for master/standby, Proxy Cluster, and Redis Cluster instances	Supported for master/standby instances

NOTE

The underlying architectures vary by Redis version. Once a Redis version is chosen, it cannot be changed. For example, you cannot upgrade a DCS Redis 3.0 instance to Redis 4.0 or 5.0. If you require a higher Redis version, buy a new instance that meets your requirements and then migrate data from the old instance to the new one.

- **Instance type**

DCS provides single-node, master/standby, Proxy Cluster, and Redis Cluster instance types. For details about their architectures and application scenarios, see [DCS Instance Types](#).

1.3 New Features of DCS for Redis 4.0

Compared with DCS for Redis 3.0, DCS for Redis 4.0 and later versions add support for the new features of open-source Redis and supports faster instance creation.

Instance deployment changed from the VM mode to physical server-based containerization mode. An instance can be created within 8 to 10 seconds.

Redis 4.0 provides the following new features:

1. New commands, such as **MEMORY** and **SWAPDB**
2. Lazyfree, delaying the deletion of large keys and reducing the impact of the deletion on system resources
3. Memory performance optimization, that is, active defragmentation

MEMORY Command

In Redis 3.0 and earlier versions, you can execute the **INFO MEMORY** command to learn only the limited memory statistics. Redis 4.0 introduces the **MEMORY** command to help you better understand Redis memory usage.

```
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

Enter **memory usage** *[key]*. If the key exists, the estimated memory used by the value of the key is returned. If the key does not exist, **nil** is returned.

```
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]>
```

NOTE

1. **usage** collects statistics on the memory usage of the value and the key, excluding the Expire memory usage of the key.
// The following is verified based on Redis 5.0.2. Results may differ in other Redis versions.
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 //After the key name length changes, the memory usage also changes. This indicates that the usage statistics contain the usage of the key.
192.168.0.66:6379> expire abc 1000000
(integer) 1
192.168.0.66:6379> memory usage abc
(integer) 60 // After the expiration time is added, the memory usage remains unchanged. This indicates that the usage statistics do not contain the expire memory usage.
192.168.0.66:6379>
2. For hashes, lists, sets, and sorted sets, the **MEMORY USAGE** command samples statistics and provides the estimated memory usage.
Usage: **memory usage** *keyset* **samples** *1000*
keyset indicates the key of a set, and *1000* indicates the number of samples.

stats

Returns the detailed memory usage of the current instance.

Usage: **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"  
... ..
```

The following table describes the meanings of some return items.

Table 1-3 MEMORY STATS return values

Return Value	Description
peak.allocated	Peak memory allocated by the allocator during Redis instance running. It is the same as used_memory_peak of info memory .
total.allocated	The number of bytes allocated by the allocator. It is the same as used_memory of info memory
startup.allocated	Initial amount of memory consumed by Redis at startup in bytes
replication.backlog	Size in bytes of the replication backlog. It is specified in the repl-backlog-size parameter. The default value is 1 MB .
clients.slaves	The total size in bytes of all replicas overheads
clients.normal	The total size in bytes of all clients overheads
overhead.total	The sum of all overheads. overhead.total is the total memory total.allocated allocated by the allocator minus the actual memory used for storing data.
keys.count	The total number of keys stored across all databases in the server
keys.bytes-per-key	Average number of bytes occupied by each key. Note that the overhead is also allocated to each key. Therefore, this value does not indicate the average key length.
dataset.bytes	Memory bytes occupied by Redis data, that is, overhead.total subtracted from total.allocated
dataset.percentage	The percentage of dataset.bytes out of the net memory usage
peak.percentage	The percentage of peak.allocated out of total.allocated
fragmentation	Memory fragmentation rate

doctor

Usage: **memory doctor**

If the value of **used_memory (total.allocated)** is less than 5 MB, **MEMORY DOCTOR** considers that the memory usage is too small and does not perform further diagnosis. If any of the following conditions is met, Redis provides diagnosis results and suggestions:

1. The peak allocated memory is greater than 1.5 times of the current **total_allocated**, that is, **peak.allocated/total.allocated** > 1.5, indicating that the memory fragmentation rate is high, and that the RSS is much larger than **used_memory**.

2. The value of high fragmentation/fragmentation is greater than 1.4, indicating that the memory fragmentation rate is high.
3. The average memory usage of each normal client is greater than 200 KB, indicating that the pipeline may be improperly used or the Pub/Sub client does not process messages in time.
4. The average memory usage of each slave client is greater than 10 MB, indicating that the write traffic of the master is too high.

purge

Usage: **memory purge**

Executes the **jemalloc** internal command to release the memory. The released objects include the memory that is occupied but not used by Redis processes, that is, memory fragments.

NOTE

MEMORY PURGE applies only to the Redis instance that uses **jemalloc** as the allocator.

Lazyfree

Problem

Redis is single-thread. When a time-consuming request is executed, all requests are queued. Before the request is completed, Redis cannot respond to other requests. As a result, performance problems may occur. One of the time-consuming requests is deleting a large key.

Principle

The Lazyfree feature of Redis 4.0 avoids the blockage caused by deleting large keys, ensuring performance and availability.

When deleting a key, Redis asynchronously releases the memory occupied by the key. The key release operation is processed in the sub-thread of the background I/O (BIO).

Usage

1. Active deletion
 - **UNLINK**
Similar to **DEL**, this command removes keys. If there are more than 64 elements to be deleted, the memory release operation is executed in an independent BIO thread. Therefore, the **UNLINK** command can delete a large key containing millions of elements in a short time.
 - **FLUSHALL** and **FLUSHDB**
An **ASYNC** option was added to **FLUSHALL** and **FLUSHDB** in order to let the entire dataset or a single database to be freed asynchronously.
2. Passive deletion: deletion of expired keys and eviction of large keys
There are four scenarios for passive deletion and each scenario corresponds to a parameter. These parameters are disabled by default.

`lazyfree-lazy-eviction no` // Whether to enable Lazyfree when the Redis memory usage reaches **maxmemory** and the eviction policy is set.

```
lazyfree-lazy-expire no // Whether to enable Lazyfree when the key with TTL is going to expire.  
lazyfree-lazy-server-del no // An implicit DEL key is used when an existing key is processed.  
slave-lazy-flush no // Perform full data synchronization for the standby node. Before loading the RDB  
file of the master, the standby node executes the FLUSHALL command to clear its own data.
```

Other New Commands

1. **swapdb**
Swaps two Redis databases.
swapdb dbindex1 dbindex2
2. **zlexcount**
Returns the number of elements in the sorted set.
zlexcount key min max

Memory and Performance Optimization

1. Compared to before, the same amount of data can be stored with less memory.
2. Used memory can be defragmented and gradually evicted.

1.4 New Features of DCS for Redis 5.0

DCS for Redis 5.0 is compatible with the new features of the open-source Redis 5.0, in addition to all the improvements and new commands in Redis 4.0.

Stream Data Structure

Stream is a new data type introduced with Redis 5.0. It supports message persistence and multicast.

Figure 1-1 shows the structure of a Redis stream, which allows messages to be appended to the stream.

Streams have the following features:

1. A stream can have multiple consumer groups.
2. Each consumer group contains a **Last_delivered_id** that points to the last consumed item (message) in the consumer group.
3. Each consumer group contains multiple consumers. All consumers share the **last_delivered_id** of the consumer group. A message can be consumed by only one consumer.
4. **pending_ids** in the consumer can be used to record the IDs of items that have been sent to the client, but have not been acknowledged.
5. For detailed comparison between stream and other Redis data structures, see **Table 1-4**.

Figure 1-1 Stream data structure

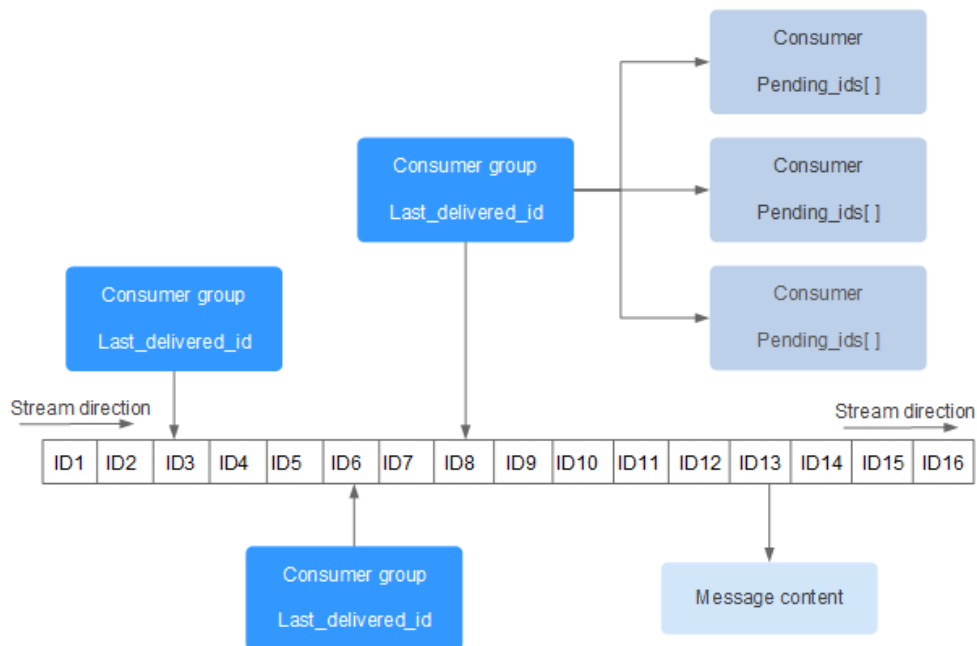


Table 1-4 Differences between streams and existing Redis data structures

Item	Stream	List, Pub/Sub, Zset
Complexity of seeking items	$O(\log(N))$	List: $O(N)$
Offset	Supported. Each item has a unique ID. The ID is not changed as other items are added or evicted.	List: Not supported. If an item is evicted, the latest item cannot be located.
Persistence	Supported. Streams are persisted to AOF and RDB files.	Pub/Sub: Not supported.
Consumer group	Supported.	Pub/Sub: Not supported.
Acknowledgment	Supported.	Pub/Sub: Not supported.
Performance	Not related to the number of consumers.	Pub/Sub: Positively related to the number of clients.
Eviction	Streams are memory efficient by blocking to evict the data that is too old and using a radix tree and listpack.	Zset consumes more memory because it does not support inserting same items, blocking, or evicting data

Item	Stream	List, Pub/Sub, Zset
Randomly deleting items	Not supported.	Zset: Supported.

Stream commands

Stream commands are described below in the order they are used. For details, see [Table 1-5](#).

1. Run the **XADD** command to add a stream item, that is, create a stream. The maximum number of messages that can be saved can be specified when adding the item.
2. Create a consumer group by running the **XGROUP** command.
3. A consumer uses the **XREADGROUP** command to consume messages.
4. After the consumption, the client runs the **XACK** command to confirm that the consumption is successful.

Figure 1-2 Stream commands

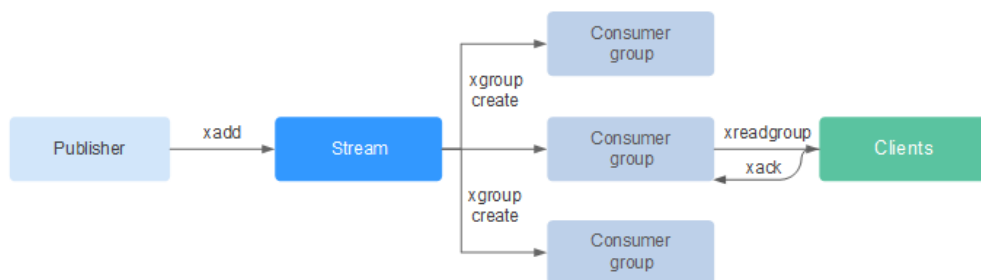


Table 1-5 Stream commands description

Command	Description	Syntax
XACK	Deletes one or multiple messages from the <i>pending entry list</i> (PEL) a consumer group of the stream.	XACK key group ID [ID ...]
XADD	Adds a specified entry to the stream at a specified key. If the key does not exist, running this command will result in a key to be automatically created based on the entry.	XADD key ID field string [field string ...]
XCLAIM	Changes the ownership of a pending message, so that the new owner is the consumer specified as the command argument.	XCLAIM key group consumer min-idle-time ID [ID ...] [IDLE ms] [TIME ms-unix-time] [RETRYCOUNT count] [FORCE] [JUSTID]

Command	Description	Syntax
XDEL	Removes the specified entries from a stream, and returns the number of entries deleted, that may be different from the number of IDs passed to the command in case certain IDs do not exist.	XDEL key ID [ID ...]
XGROUP	Manages the consumer groups associated with a stream. You can use XGROUP to: <ul style="list-style-type: none"> • Create a new consumer group associated with a stream. • Destroy a consumer group. • Remove a specified consumer from a consumer group. • Set the consumer group <i>last delivery ID</i> to something else. 	XGROUP [CREATE key groupname id-or-\$] [SETID key id-or-\$] [DESTROY key groupname] [DELCONSUMER key groupname consumername]
XINFO	Retrieves different information about the streams and associated consumer groups.	XINFO [CONSUMERS key groupname] key key [HELP]
XLEN	Returns the number of entries in a stream. If the specified key does not exist, 0 is returned, indicating an empty stream.	XLEN key
XPENDING	Obtains data from a stream through a consumer group. This command is the interface to inspect the list of pending messages in order to observe and understand what clients are active, what messages are pending to be consumed, or to see if there are idle messages.	XPENDING key group [start end count] [consumer]
XRANGE	Returns entries matching a given range of IDs.	XRANGE key start end [COUNT count]
XREAD	Reads data from one or multiple streams, only returning entries with an ID greater than the last received ID reported by the caller.	XREAD [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] ID [ID ...]
XREADGROUP P	A special version of the XREAD command, which is used to specify a consumer group to read from.	XREADGROUP GROUP group consumer [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] ID [ID ...]

Command	Description	Syntax
XREVRANGE	This command is exactly like XRANGE , but with the notable difference of returning the entries in reverse order, and also taking the start-end range in reverse order.	XREVRANGE key end start [COUNT count]
XTRIM	Trims the stream to a specified number of items, if necessary, evicting old items (items with lower IDs).	XTRIM key MAXLEN [~] count

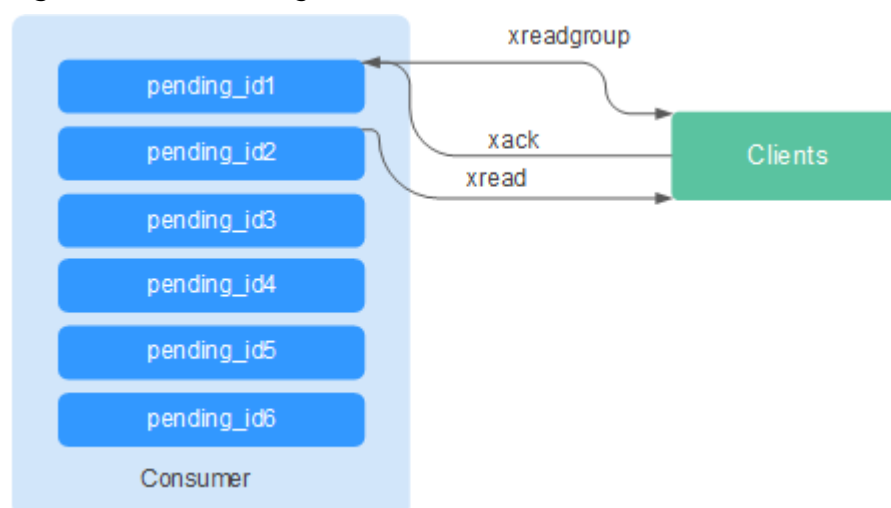
Message (stream item) acknowledgement

Compared with Pub/Sub, streams not only support consumer groups, but also message acknowledgement.

When a consumer invokes the **XREADGROUP** command to read or invokes the **XCLAIM** command to take over a message, the server does not know whether the message is processed at least once. Therefore, once having successfully processed a message, the consumer should invoke the **XACK** command to notify the stream so that the message will not be processed again. In addition, the message is removed from PEL and the memory will be released from the Redis server.

In some cases, such as network faults, the client does not invoke **XACK** after consumption. In such cases, the item ID is retained in PEL. After the client is reconnected, set the start message ID of **XREADGROUP** to 0-0, indicating that all PEL messages and messages after **last_id** are read. In addition, repeated message transmission must be supported when consumers consume messages.

Figure 1-3 Acknowledgment mechanism



Memory Usage Optimization

The memory usage of Redis 5.0 is optimized based on the previous version.

- **Active defragmentation**

If a key is modified frequently and the value length changes constantly, Redis will allocate additional memory for the key. To achieve high performance, Redis uses the memory allocator to manage memory. Memory is not always freed up to the OS. As a result, memory fragments occur. If the fragmentation ratio (**used_memory_rss/used_memory**) is greater than 1.5, the memory usage is inefficient.

To reduce memory fragments, properly plan and use cache data and standardize data writing.

For Redis 3.0 and earlier versions, memory fragmentation problems are resolved by restarting the process regularly. It is recommended that the actual cache data does not exceed 50% of the available memory.

For Redis 4.0, active defragmentation is supported, and memory is defragmented while online. In addition, Redis 4.0 supports manual memory defragmentation by running the **memory purge** command.

For Redis 5.0, improved active defragmentation is supported with the updated Jemalloc, which is faster, more intelligent, and provides lower latency.
- **HyperLogLog implementation improvements**

A HyperLogLog is a probabilistic data structure used to calculate the cardinality of a set while consuming little memory. Redis 5.0 improves HyperLogLog by further optimizing its memory usage.

For example: the B-tree is efficient in counting, but consumes a lot of memory. By using HyperLogLog, a lot of memory can be saved. While the B-tree requires 1 MB memory for counting, HyperLogLog needs only 1 KB.
- **Enhanced memory statistics**

The information returned by the **INFO** command is more detailed.

New and Better Commands

1. Enhanced client management

- redis-cli supports cluster management.

In Redis 4.0 and earlier versions, the **redis-trib** module needs to be installed to manage clusters.

Redis 5.0 optimizes redis-cli, integrating all cluster management functions. You can run the **redis-cli --cluster help** command for more information.

- The client performance is enhanced in frequent connection and disconnection scenarios.

This optimization is valuable when your application needs to use short connections.

2. Simpler use of sorted sets

ZPOPMIN and **ZPOPMAX** commands are added for sorted sets.

- ZPOPMIN key [count]

Removes and returns up to **count** members with the lowest scores in the sorted set stored at **key**. When returning multiple elements, the one with the lowest score will be the first, followed by the elements with higher scores.

- ZPOPMAX key [count]

Removes and returns up to **count** members with the highest scores in the sorted set stored at **key**. When returning multiple elements, the one with the lowest score will be the first, followed by the elements with lower scores.

3. More sub-commands added to the help command

The **help** command can be used to view help information, saving you the trouble of visiting **redis.io** every time. For example, run the following command to view the stream help information: **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 command input tips

After you enter a complete command, redis-cli displays a parameter tip to help you memorize the syntax format of the command.

As shown in the following figure, run the **zadd** command, and redis-cli displays **zadd** syntax in light color.

```
# 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 Storing LFU and LRU Information

In Redis 5.0, storage key eviction policies **LRU** and **LFU** were added to the RDB snapshot file.

- FIFO: First in, first out. The earliest stored data is evicted first.
- LRU: Least recently used. Data that is not used for a long time is evicted first.
- LFU: Least frequently used. Data that is least frequently used is evicted first.

NOTE

The RDB file format of Redis 5.0 is modified and is backward compatible. Therefore, if a snapshot is used for migration, data can be migrated from the earlier Redis versions to Redis 5.0, but cannot be migrated from the Redis 5.0 to the earlier versions.

1.5 New Features of DCS for Redis 6.0

DCS for Redis 6.0 is compatible with the new features of the open-source Redis 6.0, in addition to all the improvements and commands in Redis 5.0.

RESP3

Redis 6.0 introduced RESP3, the next-generation Redis protocol which implements more data types than RESP2.

- **Null**: a non-existent value. It replaces RESP2's *-1 and \$-1.
- **Array**: an ordered collection
- **Simple string**: a space-efficient non-binary safe string
- **Blob string**: a binary safe string
- **Simple error**: a space-efficient non-binary safe error code or message
- **Blob error**: binary safe error code or message
- **Boolean**: true or false
- **Number**: a 64-bit signed integer
- **Big number**: a large number
- **Double**: a floating point number
- **Verbatim string**: a binary safe string with a text format
- **Map**: an unordered collection of key-value pairs
- **Set**: an unordered collection of non-repeated elements
- **Attribute**: attribute key-value pairs, similar to Map
- **Push**: out-of-band data, similar to Array, used by the Redis server to push data to the client.
- **Hello**: response returned by the **HELLO** command, sent when the connection between the client and the server is established.

 **NOTE**

To use RESP3, ensure that the client SDK supports it. Otherwise, the client SDK communicates with the server through **HELLO** of RESP2.

Client-side Caching

Redis 6.0 uses tracking to proactively instruct clients to refresh their cache. There are three implementation modes:

RESP3

- Default mode
- Broadcasting mode

RESP2

- Redirecting mode

Format for enabling client-side cache tracking:

```
CLIENT TRACKING ON|OFF [REDIRECT client-id] [PREFIX prefix] [BCAST] [OPTIN][OPTOUT] [NOLOOP]
```

In RESP3, the server uses Push messages to send notifications. In the default mode, Redis remembers the keys requested by each client. When the value of a key changes, Redis sends an invalidation message to notify the corresponding client. Note that Redis notifies each client only once. If the value changes again later, the client must read the key again to enable tracking. To enable tracking in the default mode, use the following command:

```
CLIENT TRACKING ON
```

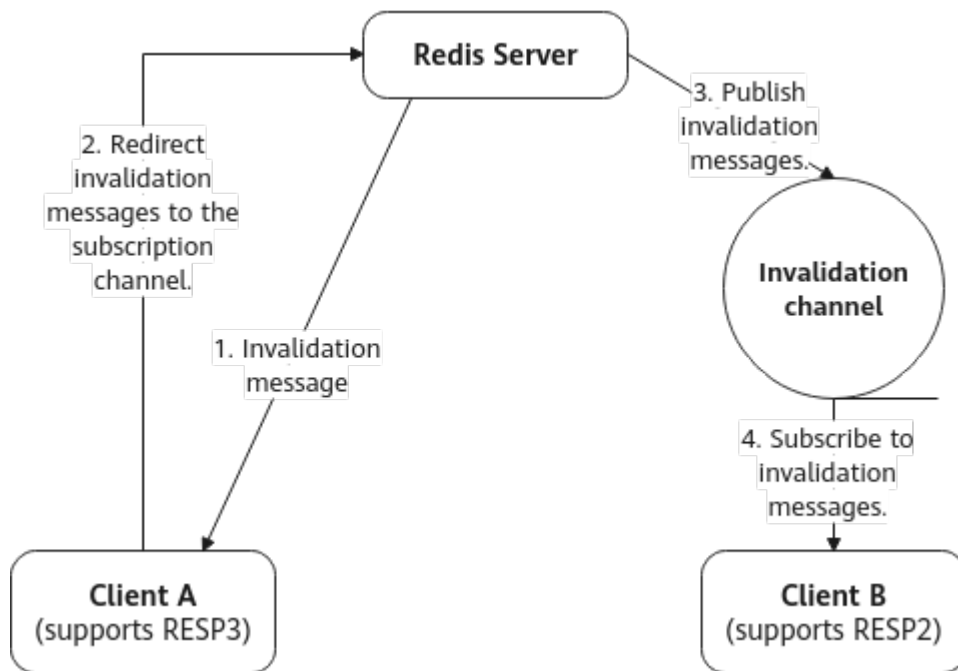
In the broadcasting mode, all clients are notified when the value of the key matching a tracked key prefix changes. If there is large number of keys that match

the key prefix or there is large number of changes, the server sends many invalidation messages, consuming much network bandwidth. To enable tracking in the broadcasting mode, use the following command:

```
CLIENT TRACKING ON BCAST PREFIX key-prefix
```

If the client SDK does not support RESP3, only the redirecting mode of RESP2 can be used to implement tracking. In this case, a dedicated client that supports RESP3 needs to act as the transit node that redirects invalidation messages from Redis to specific subscription channels.

Figure 1-4 Client cache tracking (redirecting mode)



SSL

Redis 6.0 and later versions support SSL/TLS encrypted connections and transmission. You can enable SSL on the console to generate an SSL/TLS certificate and key for an instance. When using a client to connect to the instance, specify the certificate and key. The following is a connection example:

```
redis-cli --tls --cert /etc/redis/ssl/redis.crt --key /etc/redis/ssl/redis.key --cacert /etc/redis/ssl/redis.crt
```

For details, see [Configuring SSL](#).

Faster RDB Loading

In Redis 6.0, the loading of RDB files is optimized for a 20% to 30% increase in speed.

Optimized INFO Command

The processing of the **INFO** command is optimized, especially in the scenario where a large number of clients are connected, resulting in higher performance and lower latency.

1.6 How Do I View the Version of a DCS Redis Instance?

Connect to the instance and run the **INFO** command.

Figure 1-5 Querying instance information

```
> INFO
# Server

redis_version:5.0.14

patch_version:5.0.14.1

redis_git_sha1:00000000

redis_git_dirty:0
```

1.7 What Are the Differences Between Arm-based and x86-based DCS for Redis?

DCS Redis instances support Arm and x86 CPU architectures. Arm-based Redis and x86-based Redis have the same functions and client compatibility.

x86-based Redis provides higher single-CPU performance than Arm-based Redis in scenarios that involve complex commands, such as big keys or keys whose time complexity is larger than $O(N)$.

Overall, both Arm- and x86-based Redis instances provide high performance to satisfy service requirements, but Arm-based Redis has a lower price.

NOTE

Arm-based Redis instances are unavailable in some regions. In this case, use x86-based Redis instances.

2 Feature

2.1 What Are the CPU Specifications of DCS Instances?

Redis Basic Edition

When using DCS for Redis basic edition, you only need to pay attention to critical indicators such as QPS, bandwidth, and memory. You do not need to be concerned about CPU specifications.

DCS for Redis basic edition is based on open-source Redis. Open-source Redis uses a single main thread to process commands, so only one CPU core is used on each Redis node. Increasing the memory of DCS Redis basic edition instances does not change CPU specifications.

Due to this restriction, **you can use a cluster instance and add shards to achieve higher CPU performance**. Each node in a cluster uses one vCPU by default.

Redis Professional Edition

The professional edition DCS for Redis is multi-threaded. For details about the threads, see [Table 2-1](#). Maximum CPU usage = Number of threads × 100%. For example, if number of threads is 3, maximum CPU usage is 300%.

Table 2-1 Number of threads of professional edition Redis

Instance Memory	Thread Quantity of Professional (Performance) Edition Master/Standby Instances	Thread Quantity of Professional (Storage) Edition Master/Standby Instances
8 GB	3	3
16 GB	4	6
32 GB	4	7
64 GB	4	-

2.2 What Are Shard and Replica Quantities?

Shard

A **shard** is a management unit in Redis clusters. Each shard corresponds to a redis-server process. A cluster consists of multiple shards. Each shard has multiple slots. Data is distributedly stored in the slots. Shards increase cache capacity and concurrent connections.

Each cluster instance consists of multiple shards. By default, each shard is a master/standby instance with two replicas. The number of shards is equal to the number of master nodes in a cluster instance.

Replica

A replica refers to a **node** of a DCS instance. It can be a master node or a standby node. A single-replica instance has no standby node. A two-replica instance has one master node and one standby node. For example, if the number of replicas is set to three for a master/standby instance, the instance has one master node and two standby nodes.

Number of Replicas and Shards of Different Instance Types

- **Single-node:** Each instance has only one node (one Redis process). If the Redis process is faulty, DCS starts a new Redis process for the instance.
- **Master/standby and read/write splitting:** Each instance has one shard, which contains one master node and one or more standby nodes. If the master node is faulty, master/standby switchover is triggered to restore services. The more the replicas (standby nodes), the better the reliability (performance is unaffected).
- **Cluster:** Each instance has multiple shards. By default, each shard is a master/standby instance with two replicas. For example, if a cluster instance has three shards and three replicas, each shard has three nodes (one master node and two standby nodes).

Instance Type	Shards	Replicas	Load Balancing	IP Addresses
Single-node	1	1 (only)	-	1

Instance Type	Shards	Replicas	Load Balancing	IP Addresses
Master/ Standby	1	2 (default) DCS Redis 4.0/5.0 Master/ Standby instances support 2 to 10 replicas. Master/ Standby instances of other versions support only 2 replicas.	Not supported	Same as the number of replicas
Read/write splitting	1	Default: 2; max.: 6	Supported	1
Proxy Cluster	Multiple	2 (not customizable)	Supported	1
Redis Cluster	Multiple	2 (default) DCS Redis 4.0/5.0 Redis Cluster instances support 1 to 5 replicas. Redis Cluster 6.0 instances support 1 to 2 replicas.	Not supported	Number of replicas x Number of shards

2.3 Do DCS Redis Instances Limit the Size of a Key or Value?

- The maximum allowed size of a key is 512 MB.
To reduce memory usage and facilitate key query, ensure that each key does not exceed 1 KB.
- The maximum allowed size of a string is 512 MB.
- The maximum allowed size of a Set, List, or Hash is 512 MB.
In essence, a Set is a collection of Strings; a List is a list of Strings; a Hash contains mappings between string fields and string values.

Prevent the client from constantly writing large values in Redis. Otherwise, network transmission efficiency will be lowered and the Redis server would take a longer time to process commands, resulting in higher latency.

2.4 Can I Obtain the Addresses of the Nodes in a Cluster DCS Redis Instance?

Cluster DCS Redis 3.0 instances (Proxy Cluster type) are used in the same way that you use single-node or master/standby instances. You do not need to know the backend node addresses.

For a cluster DCS Redis 4.0 or later instance (Redis Cluster type), run the **CLUSTER NODES** command to obtain node addresses:

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

In the output similar to the following, obtain the IP addresses and port numbers of all the master nodes.

```
[root@ecs-54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a 23 cluster nodes
fb75f0743af4695a3d241ff7790b2f508e4985fff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-10
40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
c16b9acaeeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
```

2.5 Does DCS for Redis Support Redis Clusters?

Yes. DCS for Redis 4.0 and 5.0 support Proxy Clusters and Redis Clusters. DCS for Redis 3.0 supports Proxy Clusters. DCS for Redis 6.0 supports Redis Clusters.

2.6 What Are Redis Data Eviction Policies?

Data is evicted from cache based on a user-defined space limit in order to make space for new data. For details, see the [Redis official website](#). You can [view or change the eviction policy](#) by configuring an instance parameter on the DCS console.

Eviction Policies Supported by DCS Redis Instances

When **maxmemory** is reached, you can select one of the following eight eviction policies:

- **noeviction**: When the memory limit is reached, DCS instances return errors to clients and no longer process write requests and other requests that could result in more memory to be used. However, **DEL** and a few more exception requests can continue to be processed.
- **allkeys-lru**: DCS instances try to evict the least recently used keys first, in order to make space for new data.
- **volatile-lru**: DCS instances try to evict the least recently used keys with an expire set first, in order to make space for new data.
- **allkeys-random**: DCS instances recycle random keys so that new data can be stored.

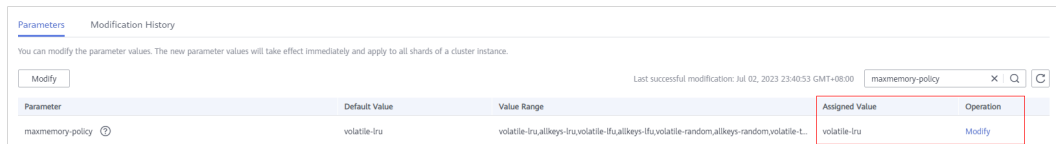
- **volatile-random**: DCS instances evict random keys with an expire set, in order to make space for new data.
- **volatile-ttl**: DCS instances evict keys with an expire set, and try to evict keys with a shorter time to live (TTL) first, in order to make space for new data.
- **allkeys-lfu**: DCS instances evict the least frequently used keys from all keys.
- **volatile-lfu**: DCS instances evict the least frequently used keys with an **expire** field from all keys.

 **NOTE**

- If no key can be recycled, **volatile-lru**, **volatile-random**, and **volatile-ttl** are the same as **noeviction**. For details, see the description of **noeviction**.
- The default eviction policy is **volatile-lru** for DCS Redis instances created in or after July 2020. The default eviction policy is **noeviction** for DCS Redis instances created before July 2020.

Viewing or Changing Eviction Policies

You can view or change the eviction policy with the **maxmemory-policy** parameter.



2.7 Does DCS for Redis Support Read/Write Splitting?

The following table describes DCS's support for read/write splitting.

Instance Type	Read/Write Splitting
Read/write splitting	Supported. NOTE To implement read/write splitting without client configurations, use read/write splitting instances .
Redis Cluster	Read/write splitting can be configured and implemented on the client. For details, see Configuration .
Master/Standby Redis 4.0, 5.0, or 6.0 basic	Read/write splitting can be implemented on a client that is able to distinguish between read and write requests.
Other	Not supported.

Configuration

- For a **Redis Cluster instance**, you can query all master and replica nodes by running the **CLUSTER NODES** command. The client will connect to replicas and configure read-only access on them.

Run the following command to query cluster nodes:

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

Read-only configuration on replicas is achieved through the **READONLY** command.

- **For a master/standby DCS Redis 4.0/5.0/6.0 basic instance**, there are two domain names displayed on the instance details page of the console: a read/write address (master node) and a read-only address (standby node). On the client, you can direct write requests to the read/write domain name and read requests to the read-only domain name.
- For a **read/write splitting instance**, read/write splitting is implemented on the server side by default. Proxies distinguish between read and write requests, and forward write requests to the master node and read requests to the standby node. You do not need to perform any configuration on the client.

2.8 Does DCS for Redis Support Multi-DB?

DCS's support for multiple databases (multi-DB) is as follows:

- Single-node, read/write splitting, and master/standby DCS Redis instances: Multi-DB is supported. By default, there are 256 databases, numbering 0–255. The default database is DB0. Multi-DB is used for data isolation. The size of each database is not evenly allocated. As a result, one database may fully occupy the memory of the instance.
- Proxy Cluster: There is only one database by default.
 - For details about how to create a Proxy Cluster instance with multiple databases, see [How Do I Create a Multi-DB Proxy Cluster Instance?](#)
 - For details about how to enable multi-DB for a single-DB Proxy Cluster instance, see [What Are the Constraints on Implementing Multiple Databases on a Proxy Cluster Instance?](#)

NOTE

Proxy Cluster DCS Redis 3.0 instances do not support multi-DB.

- Redis Cluster DCS instances: Multi-DB is not supported. There is only one database (DB0).

The number of databases cannot be changed, and the size of each database cannot be customized.

2.9 Does DCS Support External Extensions, Plug-ins, or Modules?

No. DCS for Redis does not support external extensions, plug-ins, or modules. There is no plan for supporting modules.

2.10 Does DCS Support Data Persistence? What Is the Impact of Persistence?

Support for Persistence

- DCS Redis instances:
 - Single-node: Data persistence is not supported.
 - Master/standby, read/write splitting, and cluster (except single-replica clusters): Data persistence is supported.
- DCS Memcached instances:
 - Single-node: Data persistence is not supported.
 - Master/standby: Data persistence is supported.

Persistence Modes

- DCS supports only AOF persistence by default. You can enable or disable persistence as required. All instances except single-node and single-replica cluster ones are created with AOF persistence enabled.
- DCS does not support RDB persistence by default and you cannot configure the **save** parameter. If RDB persistence is required for a master/standby or cluster instance of Redis 4.0 or later, you can use the backup and restoration function to back up the instance data to an RDB file and store the data in OBS.

Disk Used for Persistence

For DCS Redis 4.0 and later instances, data is persisted to SSD disks.

Impact of AOF Persistence

After AOF persistence is enabled, the Redis-Server process needs to record operations in the AOF file for data persistence.

- If the disk or I/O of the underlying compute node is faulty, the latency may increase or a master/standby switchover may occur.
- Redis-Server periodically rewrites the AOF. During a rewrite, the latency may be high for a short time. For details about the AOF rewriting rules, see [When Will AOF Rewrites Be Triggered?](#)

If DCS instances are used for application acceleration, you are advised to disable AOF persistence for higher performance and stability.

Exercise caution when disabling AOF persistence. After it is disabled, cached data may be lost in extreme scenarios, for example, when both the master and standby nodes are faulty.

To disable AOF persistence, set parameter **appendonly** to **no** on the instance details page.

Can I Enable Persistence Only on Replicas and Not on Masters?

Yes. For a master/standby or cluster DCS Redis 4.0/5.0/6.0 basic instance, you can set the **appendonly** parameter to **only-replica** to enable persistence only on replicas.

This function is not available for other instance versions or types.

NOTE

- By default, the **appendonly** parameter has only two options: **yes** and **no**. To set it to **only-replica**, contact O&M personnel.
- Without writing and rewriting AOF on the master node, persistence only on replicas improves performance but lowers reliability, compared with persistence on both master and replica nodes. Select a mode based on service requirements.

2.11 Do DCS Redis Instances Limite Data Quantity?

No, but the data volume cannot exceed the memory of a DCS instance.

3 Security

3.1 How Do I Configure a Security Group?

DCS Redis 3.0/4.0/5.0/6.0 and Memcached instances are deployed in different modes. Therefore, the access control method varies.

- To control access to DCS Redis 3.0, Memcached, and Redis 6.0 professional edition instances, you can use security groups. Whitelists are not supported. Security group operations are described in this section.
- To control access to DCS Redis 4.0/5.0/6.0 basic edition instances, you can use whitelists. Security groups are not supported. Whitelist operations are described in [Managing IP Address Whitelist](#).

The following describes how to configure security groups for **intra-VPC access** and **public access** to DCS Redis 3.0, Memcached, and Redis 6.0 professional edition instances.

Intra-VPC Access to DCS Redis 3.0, Memcached, and Redis 6.0 Professional Edition Instances

To prevent cross-VPC access from increasing latency and affecting DCS instance performance, you can deploy your client on an ECS that is in the same VPC and subnet as the DCS instance.

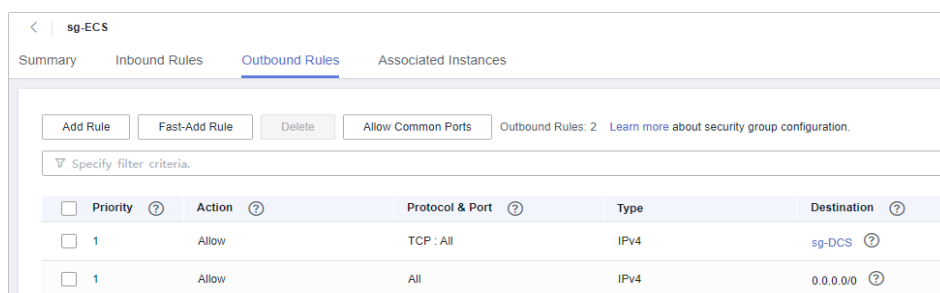
In addition, you must configure correct rules for the security groups of both the ECS and DCS instance so that you can access the instance through your client.

- If the ECS and DCS instance are configured with the same security group, network access in the group is not restricted by default.
- If the ECS and DCS instance are configured with different security groups, add security group rules to ensure that the ECS and DCS instance can access each other.

 NOTE

- Suppose that the ECS on which the client runs belongs to security group **sg-ECS**, and the DCS instance that the client will access belongs to security group **sg-DCS**.
 - The following uses port 6379 for DCS Redis 3.0 instances as an example. For other instances, use the actual port.
 - The remote end is a security group or an IP address.
- a. Configuring security group for the ECS.

Add the following outbound rule to allow the ECS to access the DCS instance. Skip this rule if there are no restrictions on the outbound traffic.

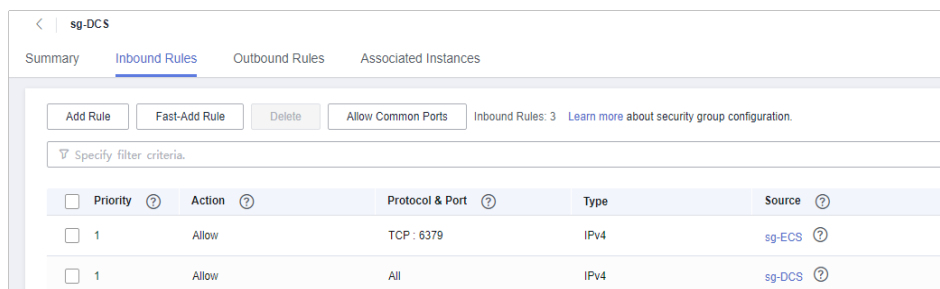


The screenshot shows the 'Outbound Rules' tab for security group 'sg-ECS'. It displays two rules:

Priority	Action	Protocol & Port	Type	Destination
1	Allow	TCP : All	IPv4	sg-DCS
1	Allow	All	IPv4	0.0.0.0/0

- b. Configuring security group for the DCS instance.

To ensure that your client can access the DCS instance, add the following inbound rule to the security group configured for the DCS instance:



The screenshot shows the 'Inbound Rules' tab for security group 'sg-DCS'. It displays two rules:

Priority	Action	Protocol & Port	Type	Source
1	Allow	TCP : 6379	IPv4	sg-ECS
1	Allow	All	IPv4	sg-DCS

NOTICE

For the source IP address, use the specified IP address of the DCS instance. Avoid using **0.0.0.0/0** to prevent ECSs bound with the same security group from being attacked by Redis vulnerability exploits.

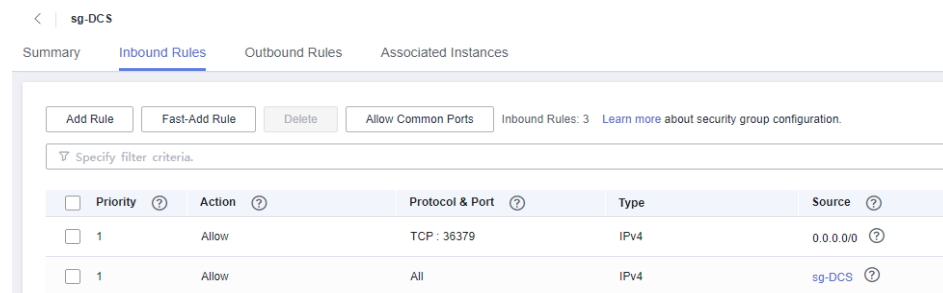
Public Access to DCS Redis 3.0 Instances

A client can access a DCS instance only after rules are correctly configured for the security group of the instance.

For example, for security group **sg-DCS**, you need to configure the following rules in the inbound direction:

Set the protocol to TCP and source IP address to 0.0.0.0/0 or a specified client address. If SSL is enabled, set the port number to 36379. If SSL is disabled, set the port number to 6379. See the following figure.

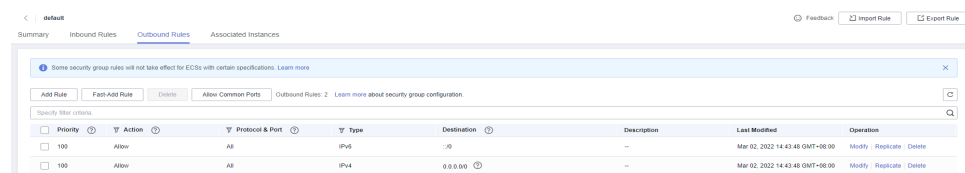
Figure 3-1 Security group rule (port 36379 is used for the example)



Security Group of a Migration Task

- When creating an online migration task, select a security group. Its outbound rules must allow traffic over the IP addresses and ports of the source and target Redis instances. By default, all outbound traffic is allowed.
- Backup import uses the **default** security group. Ensure that all outbound traffic is allowed (this is the default setting.)

Figure 3-2 Outbound rules of the migration security group



3.2 Why Can't Security Groups Be Configured for DCS Redis 4.0/5.0/6.0 Basic Edition Instances?


Currently, DCS Redis 4.0/5.0/6.0 basic edition instances use VPC endpoints and do not support security groups.

To allow access only from specific IP addresses to a DCS Redis 4.0, 5.0, or 6.0 basic edition instance, add the IP addresses to the instance whitelist.

If no whitelists are added for the instance or the whitelist function is disabled, all IP addresses that can communicate with the VPC can access the instance.

Creating a Whitelist Group

Step 1 Log in to the [DCS console](#).

Step 2 Click  in the upper left corner of the management console and select the region where your instance is located.

Step 3 In the navigation pane, choose **Cache Manager**.

Step 4 Click the name of a DCS instance.

Step 5 Choose **Instance Configuration > Whitelist**. On the displayed page, click **Create Whitelist Group**.

Step 6 In the **Create Whitelist Group** dialogue box, specify **Group Name** and **IP Address/Range**.

Table 3-1 Whitelist parameters

Parameter	Description	Example
Group Name	Whitelist group name of the instance. A maximum of four whitelist groups can be created for each instance.	DCS-test
IP Address/Range	A maximum of 20 IP addresses or IP address ranges can be added to an instance. Separate multiple IP addresses or IP address ranges with commas. Unsupported IP address and IP address range: 0.0.0.0 and 0.0.0.0/0.	10.10.10.1,10.10.10.10

Step 7 Click **OK**.

The whitelist function takes effect immediately after the whitelist group is created. Only whitelisted IP addresses can access the instance. For persistent connections, the whitelist takes effect after reconnection.

 **NOTE**

- In the whitelist group list, click **Edit** to modify the IP addresses or IP address ranges in a group, and click **Delete** to delete a whitelist group.
- After whitelist has been enabled, you can click **Disable Whitelist** above the whitelist group list to allow all IP addresses connected to the VPC to access the instance.

----End

3.3 How Can I Secure My DCS Redis Instances?

Redis is one of the most powerful and widely used open-source cache technologies. However, the open-source Redis does not have robust security features of its own. It is vulnerable to malicious Internet attacks, possibly causing data breaches.

To secure your DCS Redis instances, consider taking the following advice:

- Network connection configurations

- a. Encrypt sensitive data and disable public access.
Sensitive data must be encrypted before being stored. Do not use public access unless otherwise required.
 - b. Configure access rules for your security groups.
Security groups and VPCs are designed for securing network access. Allow access over as few ports as possible to avoid risks.
 - c. Configure ECS firewalls.
Configure firewall filtering rules for the ECS where your client runs.
 - d. Set the instance password.
 - e. Configure a whitelist.
- redis-cli usage
 - a. Hide the password.
Problem: If the **-a <password>** option is used, the password may show up when the **ps** command is run.
Solution: Modify the Redis source code. Hide the password immediately after starting redis-cli by calling the main function.
 - b. Disable sudo in running scripts.
Problem: Parameters for starting redis-cli contain sensitive patterns related to the password, which may show up when the **ps** command is run and may be logged.
Solution: Access the instance by calling APIs (or through redis-py in Python). Do not allow switching to the **dbuser** user using sudo in redis-cli.

3.4 Does DCS for Redis Support SSL Encrypted Transmission?

By default, SSL is disabled for DCS Redis 6.0 basic edition instances. To enable it, see .

For public access to DCS instances (supported only by DCS Redis 3.0 instances), you can enable TLS encryption with Stunnel. For details, see the [instructions on installing and configuring Stunnel](#). When DCS provisions instances, the specified Certificate Chain (CA) will generate a unique service certificate for each instance. When connecting to an instance, clients can use the CA root certificates downloaded from the management console to authenticate the instance server and encrypt data during transmission.

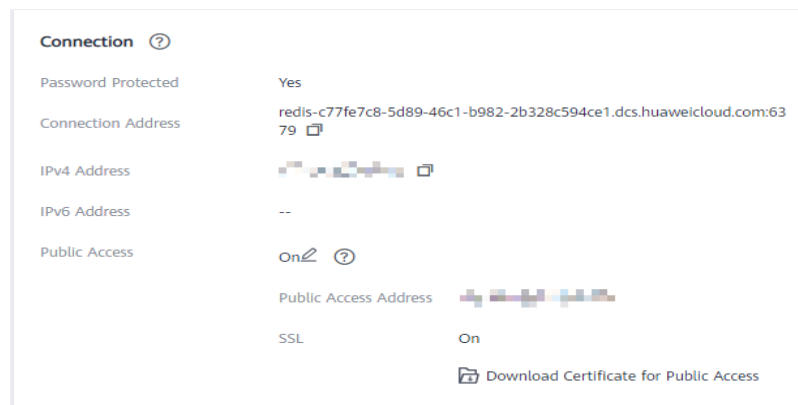
DCS Redis for 4.0/5.0 only support plaintext transmission. They do not support SSL encrypted transmission.

3.5 How Do I Enable or Disable SSL for Public Access to a DCS Redis 3.0 Instance?

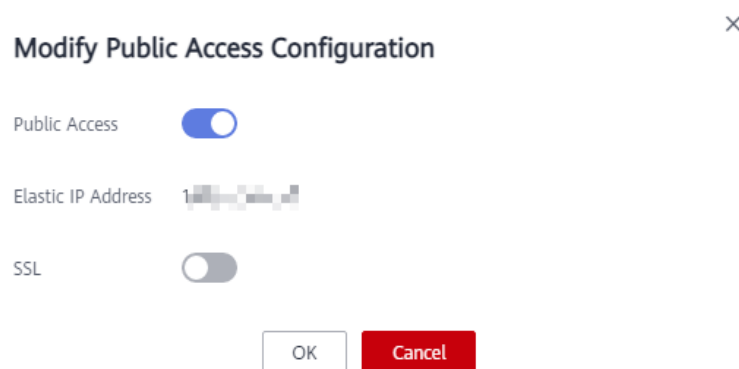
When you enable public access, SSL is enabled by default.

To disable SSL encryption, perform the following steps:

1. Open the page for configuring public access.



2. Disable SSL encryption, and click **OK**.



3. In the **Connection** area on the instance details page, **SSL** is disabled.

3.6 Does DCS Support Cross-AZ Deployment?

Master/standby, read/write splitting, and cluster DCS Redis instances and master/standby DCS Memcached instances can be deployed across availability zones (AZs).

- If instance nodes in an AZ are faulty, nodes in other AZs will not be affected. The standby node automatically becomes the master node to continue to operate, ensuring disaster recovery (DR).
- Cross-AZ deployment does not compromise the speed of data synchronization between the master and standby nodes.

3.7 Is a Password Required for Accessing an Instance? How Do I Set a Password?

- A DCS Redis instance can be access with or without a password. You can directly access a DCS Redis instance through a Redis client without setting a password. However, for security purposes, you are advised to set a password for authentication and verification whenever possible. The password must be set when you create the instance.

- A DCS Memcached instance can be access with or without a password. You can select any Memcached client that supports the Memcached text protocol and binary protocol based on specific application features. The password must be set when you create the instance.
- To change the Redis instance access mode, or change or reset a password, see [Managing Passwords](#).

3.8 What Is Sentinel?

Overview

High availability in Redis is implemented through Sentinel. Sentinel helps you defend against certain types of faults without manual intervention, and complete tasks such as monitoring, notification, and client configuration. For details, see the [Redis official website](#).

Principles

Redis Sentinel is a distributed system where multiple Sentinel processes work together. It has the following advantages:

1. Fault detection is performed only when multiple Sentinels agree that a master node is unavailable, which reduces the possibility of false positives.
2. Even if some Sentinel processes are faulty, the Sentinel system can still work properly to prevent faults.

On a higher level, there is a larger distributed system consisting of Sentinels, Redis master and replica nodes, and clients connected to Sentinels and Redis.

Functions

- Monitoring: Sentinel continuously checks whether the master and replica nodes are working properly.
- Notification: If a node is faulty, Sentinel can notify the system administrator or other computer programs by calling an API.
- Automatic failover: If the master node is abnormal, Sentinel starts a failover to promote a replica to master. Other replicas replicate data from the new master node. Applications that use the Redis instance will be notified that they should connect to the new address.
- Client configuration: Sentinel serves as the authoritative source for client service discovery. Clients connect to Sentinel and requests the address of the master Redis node that is responsible for specific services. If a failover occurs, Sentinels delivers the new address.

3.9 Does DCS Manage Instances in Sentinel Mode?

- Cluster instances, master/standby and read/write splitting DCS Redis 4.0, 5.0, and 6.0 instances support Sentinels. Sentinels monitor the running status of both the master and standby nodes of a master/standby instance and each shard of a cluster instance. If the master node becomes faulty, a failover will be performed and the service will be resumed automatically.

Sentinels are invisible to you and is used only in the service.

- DCS for Redis 3.0 does not support Redis Sentinel. Instead, it uses keepalive to monitor master and replica nodes and to manage failovers automatically.

4 Client and Network Connection

4.1 Does DCS Support Public Access?

- Redis 3.0

Currently, **public access is supported only by password-protected DCS Redis 3.0 instances**. You can enable or disable SSL for public access. You are advised to download a CA certificate in advance and use it to verify the certificate of a DCS instance for security purposes. For more information, see [Public Access to a DCS Redis Instance](#).

- Redis 4.0 and later

Public access is not supported by DCS Redis 4.0, 5.0, and 6.0 instances. If public access is required for a single-node, master/standby, or Proxy Cluster instance, use Nginx to redirect connections through an ECS configured with the same VPC and security group as the DCS instance. For details, see [Using Nginx for Public Access to Single-node, Master/Standby, or Proxy Cluster DCS Redis Instances](#).

Redis Cluster instances cannot be accessed using Nginx over public networks. You can use Elastic Load Balance (ELB) to access different types of DCS instances over public networks. For details, see [Using ELB for Public Access to DCS](#).

- Memcached

Public access is not supported. The ECS that serves as a client and the DCS instance that the client will access must belong to the same VPC. In the application development and debugging phases, you can also use SSH to access your instance in the local environment. For details, see [Using SSH Tunneling for Public Access to a DCS Instance](#).

4.2 Troubleshooting Redis Connection Failures

Overview

This topic describes why Redis connection problems occur and how to solve the problems.

Problem Classification

To troubleshoot abnormal connections to a Redis instance, check the following items:

- [Connection Between the Redis Instance and the ECS](#)
- [Public Access \(Redis 3.0 Only\)](#)
- [Password](#)
- [Instance Configuration](#)
- [Client Connections](#)
- [Bandwidth](#)
- [Redis Performance](#)

Connection Between the Redis Instance and the ECS

The ECS where the client is located must be in the same VPC as the Redis instance and be able to communicate with the Redis instance.

- For a Redis 3.0 or professional edition instance, check the security group rules of the instance and the ECS.
Correctly configure security group rules for the ECS and the Redis instance to allow the Redis instance to be accessed. For details, see [How Do I Configure a Security Group?](#)
- For a DCS Redis 4.0/5.0/6.0 basic instance, check the whitelist of the instance.
If the instance has a whitelist, **ensure that the client IP address is included in the whitelist**. Otherwise, the connection will fail. For details, see [Managing IP Address Whitelist](#). If the client IP address changes, add the new IP address to the whitelist.
- Check the regions of the Redis instance and the ECS.
If the Redis instance and the ECS are not in the same region, create another Redis instance in the same region as the ECS and migrate data from the old instance to the new instance by referring to [Data Migration Guide](#).
- Check the VPCs of the Redis instance and the ECS.
Different VPCs cannot communicate with each other. An ECS cannot access a Redis instance if they are in different VPCs. You can establish VPC peering connections to allow the ECS to access the Redis instance across VPCs.
For more information on how to create and use VPC peering connections, see [VPC Peering Connection](#).

Public Access (Redis 3.0 Only)

Before accessing a Redis instance through a public network, ensure that the instance supports public access. For details, see the [public access explanation](#).

- **Symptom:** "Error: Connection reset by peer" is displayed or a message is displayed indicating that the remote host forcibly closes an existing connection.
 - **Possible cause 1:** The security group is incorrectly configured.
Solution: Correctly configure the Redis instance and access the instance by following the [public access instructions](#).

- **Possible cause 2:** Check whether the VPC subnet where Redis resides is associated with a network ACL and whether the network ACL denies outbound traffic. If yes, remove the ACL restriction.
- **Possible cause 3:** SSL encryption has been enabled, but Stunnel is not configured during connection. Instead, the IP address displayed on the console was used for connection.

Solution: When enabling SSL encryption, install and configure the Stunnel client. For details, see [Enabling SSL Encryption](#). In the command for connecting to the Redis instance, the address must be set to the IP address and port number of the Stunnel client. Do not use the public access address and port displayed on the console.

- **Symptom:** Public access has been automatically disabled.

Cause: The EIP bound to the DCS Redis instance is unbound. As a result, public access is automatically disabled.

Solution: Enable public access for the instance and bind an EIP to the instance on the management console. Then, try again.

Password

If the instance password is incorrect, the port can still be accessed but the authentication will fail. If you forget the password, you can reset the password. For details, see [Resetting Instance Passwords](#).

Instance Configuration

If a connection to Redis is rejected, log in to the DCS console, go to the instance details page, and modify the **maxclients** parameter. For details, see [Modifying Configuration Parameters](#).

Client Connections

- The connection fails when you use redis-cli to connect to a Redis Cluster instance.

Solution: Check whether **-c** is added to the connection command. Ensure that the correct connection command is used when connecting to the cluster nodes.

- Run the following command to connect to a Redis Cluster instance:

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

- Run the following command to connect to a single-node, master/standby, or Proxy Cluster instance:

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

For details, see [Access Using redis-cli](#).

- Error "Read timed out" or "Could not get a resource from the pool" occurs.

Solution:

- Check if the **KEYS** command has been used. This command consumes a lot of resources and can easily block Redis. Instead, use the **SCAN** command and do not execute the command frequently.
- Check if the DCS instance is Redis 3.0. Redis 3.0 uses SATA disks. During AOF persistence, the disk performance may occasionally deteriorate and

cause a connection failure. In this case, disable AOF persistence if data persistence is not required. Alternatively, you can use a DCS Redis 4.0 instance or later because they use SSD disks that offer higher performance.

- Error "unexpected end of stream" occurs and causes service exceptions.

Solution:

- Optimize the Jedis connection pool by referring to [Recommended Jedis Parameter Settings](#).
- Check whether there are many big keys. For details, see [How Do I Avoid Big Keys and Hot Keys?](#)

- The connection is interrupted.

Solution:

- Modify the application timeout duration.
- Optimize the service to avoid slow queries.
- Replace the **KEYS** command with the **SCAN** command.

- If an error occurs when you use the Jedis connection pool, see [Troubleshooting a Jedis Connection Pool Error](#).

Bandwidth

If the bandwidth reaches the upper limit of the corresponding instance specifications, Redis connections may time out.

You can view the **Flow Control Times** metric to check whether the bandwidth has reached the upper limit.

Then, check whether the instance has big keys and hot keys. If a single key is too large or overloaded, operations on the key may occupy too many bandwidth resources. For details about big keys and hot keys, see [Analyzing Big Keys and Hot Keys](#).

You can [temporarily expand the bandwidth](#) of a DCS Redis 4.0 or later instance on the console. The bandwidth expansion is valid for seven days.

Redis Performance

Connections to an instance may become slow or time out if the CPU usage spikes due to resource-consuming commands such as **KEYS**, or too much memory is used because the expiration time is not set for the instance or expired keys remain in the memory. In these cases, do as follows:

- Use the **SCAN** command instead of the **KEYS** command, or disable the **KEYS** command.
- Check the monitoring data and configure alarm rules. For details, see [Setting Alarm Rules for Critical Metrics](#).

For example, you can view the **Memory Usage** and **Used Memory** metrics to keep track of the instance memory usage, and view the **Connected Clients** metric to determine whether the instance connections limit has been reached.

- Check whether the instance has big keys and hot keys.

For details about the operations of big key and hot key analysis, see [Analyzing Big Keys and Hot Keys](#).

4.3 Does DCS Support Cross-VPC Access?

Cross-VPC means the client and the instance are not in the same VPC.

The following assumes that public access is disabled for a DCS instance. Generally, VPCs are isolated from each other and an ECS cannot access a DCS instance that belongs to a different VPC from the ECS.

You can establish VPC peering connections to allow the ECS to access the DCS instance across VPCs.

When using VPC peering connections to access DCS instances across VPCs, adhere to the following rules:

- If CIDR Blocks 172.16.0.0/12 to 172.16.0.0/24 are used during DCS instance creation, the client cannot be in any of the following CIDR Blocks: 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24.
- If CIDR Blocks 192.168.0.0/16 to 192.168.0.0/24 are used during DCS instance creation, the client cannot be in any of the following CIDR Blocks: 172.31.1.0/24, 172.31.2.0/24, and 172.31.3.0/24.
- If CIDR Blocks 10.0.0.0/8 to 10.0.0.0/24 are used during DCS instance creation, the client cannot be in any of the following CIDR Blocks: 172.31.1.0/24, 172.31.2.0/24, and 172.31.3.0/24.

For more information on how to create and use VPC peering connections, see [VPC Peering Connection](#).

4.4 Will I Be Charged for the EIP Used for Public Access to a DCS Redis Instance?

Yes. You must pay for the EIP used for public access to a DCS Redis 3.0 instance.

Before enabling public access, you must have an available EIP. For the billing details, see the [EIP pricing details](#).

4.5 Why Is "(error) NOAUTH Authentication required" Displayed When I Access a DCS Redis Instance?

This is because you have enabled password-free access for the instance. To prevent the error message from appearing, do not enter any password.

4.6 What Should I Do If Access to DCS Fails After Server Disconnects?

Analysis: If persistent connections ("pconnect" in Redis terminology) or connection pooling is used and connections are closed after being used for connecting to DCS instances, errors will be returned at attempts to reuse the connections.

Solution: When using pconnect or connection pooling, do not close the connection after the end of a request. If the connection is dropped, re-establish it.

4.7 Why Do Requests Sometimes Time Out in Clients?

Occasional timeout errors are normal because of network connectivity and client timeout configurations.

You are advised to include reconnection operations into your service code to avoid service failure if a single request fails.

When a connection times out, check whether AOF persistence is enabled for Redis. Then, determine whether to enable the function (learn about the [impact of AOF persistence](#)) as required. Disabling AOF persistence improves stability and reduces blocking on the client to prevent connection failures.

If timeout errors occur frequently, contact technical support.

4.8 What Should I Do If an Error Is Returned When I Use the Jedis Connection Pool?

The error message that will possibly be displayed when you use the Jedis connection pool JedisPool is as follows:

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

If this error message is displayed, check whether your instance is running properly. If it is running properly, perform the following checks:

Step 1 Check the network.

1. Check the IP address configurations.

Check whether the IP address configured on the Jedis client is the same as the subnet address configured for your DCS instance. If public access is enabled for your instance, check whether the IP address configured on the Jedis client is the same as the EIP bound to your instance. If they are inconsistent, modify the IP address configuration and then try again.

2. Test the network.

Use the ping command and telnet on the client to test the network.

– If the network cannot be pinged:

- For intra-VPC access, ensure that the client and your DCS instance are in the same VPC, and [security group rules or whitelist](#) has been correctly configured.
- For public access with SSL, ensure that you have configured the security group of your DCS instance, allowing access through port 36379 as instructed in [Security Group Configurations](#).
- For public access without SSL, ensure that you have configured the security group of your DCS instance, allowing access through port 6379 as instructed in [Security Group Configurations](#).

- If the IP address can be pinged but telnet failed, restart your instance. If the problem persists after the restart, contact technical support.

Step 2 Check the number of connections.

Check whether the number of established network connections exceeds the upper limit configured for JedisPool. If the number of established connections approaches the configured upper limit, restart the DCS service and check whether the problem persists. If the number of established connections is far below the upper limit, continue with the following checks.

In Unix or Linux, run the following command to query the number of established network connections:

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

In Windows, run the following command to query the number of established network connections:

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

Step 3 Check the JedisPool code.

If the number of established connections approaches the upper limit, determine whether the problem is caused by service concurrency or incorrect usage of JedisPool.

When using JedisPool, you must call `jedisPool.returnResource()` or `jedis.close()` (recommended) to release the resources after you call `jedisPool.getResource()`.

Step 4 Check the number of TIME_WAIT connections.

Run the `ss -s` command to check whether there are too many **TIME_WAIT** connections on the client.

```
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
```

If there are too many **TIME_WAIT** connections, modify the kernel parameters by running the `/etc/sysctl.conf` command as follows:

```
##Uses cookies to prevent some SYN flood attacks when the SYN waiting queue overflows.
net.ipv4.tcp_syncookies = 1
##Reuses TIME_WAIT sockets for new TCP connections.
net.ipv4.tcp_tw_reuse = 1
##Enables quick reclamation of TIME_WAIT sockets in TCP connections.
net.ipv4.tcp_tw_recycle = 1
##Modifies the default timeout time of the system.
net.ipv4.tcp_fin_timeout = 30
```

After the modification, run the `/sbin/sysctl -p` command for the modification to take effect.

Step 5 If the problem persists after you perform the preceding checks, perform the following steps.

Capture packets and send packet files along with the time and description of the exception to technical support for analysis.

Run the following command to capture packets:

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

In Windows, you can also install the Wireshark tool to capture packets.

NOTE

For public access, change the port number to **36379**.
Replace the NIC name to the actual one.

----End

4.9 How Do I Access a DCS Redis Instance Through Redis Desktop Manager?

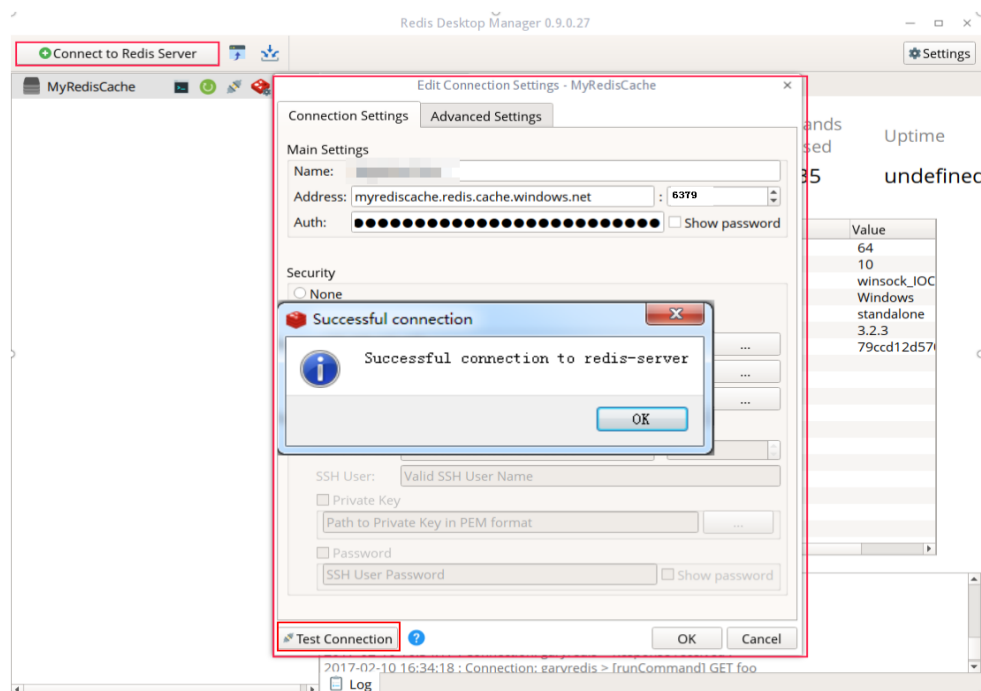
You can access a DCS Redis 3.0 instance through the Redis Desktop Manager within a VPC or over the Internet.

Within a VPC

1. Enter the address, port number (6379), and authentication password of the DCS instance you want to access.
2. Click **Test Connection**.

The system displays a success message if the connection is successful.

Figure 4-1 Accessing a DCS Redis instance through Redis Desktop Manager over the intranet



 **NOTE**

When accessing a cluster DCS instance, the Redis command is run properly, but an error message may display on the left because DCS clusters are based on Codis, which differs from the native Redis in terms of the **INFO** command output.

Over the Internet

Check whether SSL is enabled for the DCS instance you want to access.

- If SSL is not enabled, enter the public access address of the instance.
Configure the inbound rule of the security group of the instance, allowing access over port 6379.
- If SSL is enabled, install the Stunnel client and then connect to the Redis server through Redis Desktop Manager. Note:
 - The Stunnel client must be installed. For details about how to install and configure the Stunnel client, see [Stunnel instructions](#).
 - The address must be set to **127.0.0.1** rather than the public IP address. Otherwise, "connection reset" will be returned.

When SSL is enabled, Redis is accessed through an encrypted channel established by Stunnel. After a request is sent from Redis Desktop Manager to the listening port of 127.0.0.1, the request is encrypted and sent to the Redis instance through port 36379 over a public network.

Configure the inbound rule of the security group of the instance, allowing access over port 36379.

To enable SSL, disable public access first. Then, enable SSL while re-enabling public access. To disable SSL, disable public access first. Then, disable SSL while re-enabling public access.

4.10 What If "ERR Unsupported CONFIG subcommand" is Displayed in SpringCloud?

By using DCS Redis instances, Spring Session can implement session sharing. When interconnecting with Spring Cloud, the following error information is displayed:

Figure 4-2 Spring Cloud error information

```

org.springframework.util.ReflectionUtils$MethodAndOverrideChecker: Invocation of this method failed: nested exception is org.springframework.dao.InvalidDataAccessApiUsageException: ERR Unsupported CONFIG subcommand
dis.clients.jedis.exceptions.JedisDataException: ERR Unsupported CONFIG subcommand
2019-02-01 00:36:59 INFO com.alibaba.druid.pool.DruidDataSource - {dataSource-2} closed
2019-02-01 00:36:59 INFO com.alibaba.druid.pool.DruidDataSource - {dataSource-1} closed
2019-02-01 00:36:59 ERROR org.springframework.web.context.ContextLoader - Context initialization failed
org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'enableRedisKeyspaceNotificationsInitializer' defined in class path resource [org.springframework.session/data/redis/annotations/web/http/RedisHttpSessionConfiguration.class]: Invocation of init-method failed; nested exception is org.springframework.dao.InvalidDataAccessApiUsageException: ERR Unsupported CONFIG subcommand
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1784)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:513)
    at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:482)
    at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:512)
    at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:228)
    at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:510)
    at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:280)
    at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:756)
    at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:862)
    at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:450)
    
```

For security purposes, DCS does not support the **CONFIG** command initiated by a client. You need to perform the following steps:

1. On the DCS console, set the value of the **notify-keyspace-event** parameter to **Egx** for a DCS Redis instance.

2. Add the following content to the XML configuration file of the Spring framework:

```
<util:constant  
static-  
field="org.springframework.session.data.redis.config.ConfigureRedisAction.NO_  
OP"/>
```
3. Modify the related Spring code. Enable the **ConfigureRedisAction.NO_OP** bean component to forbid a client to invoke the **CONFIG** command.

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

For more information, see the [Spring Session Documentation](#).

NOTICE

Session sharing is supported only by **single-node** and **master/standby** DCS Redis instances, but not by cluster DCS Redis instances.

4.11 What Can I Do If I Fail to Access a DCS Instance Using Its Domain Name Address?

If a client fails to connect to a DCS instance using the domain name address, set the DNS server address of the subnet to the private DNS server address.

For details, see [How Do I Switch to a Private DNS Server?](#)

4.12 Can I Access DCS Instances in a Local Environment?

- If public access is disabled for a DCS instance, you cannot access it in local environments and can only access it through an ECS in a VPC that can communicate with the instance. VPCs are used to ensure network security of services.
You can connect to a DCS instance from your local environment by using an ECS that can communicate with your instance for forwarding your requests. For details, see [Using SSH Tunneling for Public Access to a DCS Instance](#).
- If public access is enabled, DCS instances can be accessed in local environments. For more information, see [Public Access to a DCS Redis Instance](#).

4.13 What Should Be Noted When Using Redis for Pub/Sub?

The [Redis official website](#) describes Pub/Sub in detail. When using Redis for Pub/Sub, note the following:

- Your client must process messages in a timely manner.
Your client subscribes to a channel. If it does not receive messages in a timely manner, DCS instance messages may be overstocked. If the size of accumulated messages reaches the threshold (32 MB by default) or remains at a certain level (8 MB by default) for a certain period of time (1 minute by default), your client will be automatically disconnected to prevent server memory exhaustion.
- Your client must support connection re-establishment in case of disconnection.
In the event of a disconnection, you need to run the **subscribe** or **psubscribe** command on your client to subscribe to a channel again. Otherwise, your client cannot receive messages.
- Do not use pub/sub in scenarios with high message reliability requirements.
The Redis pub/sub is not a reliable messaging system. Messages that are not retrieved will be discarded when your client is disconnected or a master/standby switchover occurs.

4.14 Why Is Public Access of My DCS Redis Instance Unintentionally Disabled?

Symptom: Public access has been enabled for a DCS Redis 3.0 instance but is suddenly disabled.

Cause: The EIP bound to the DCS Redis instance is unbound. As a result, public access is automatically disabled.

4.15 What Can I Do If Error "Cannot assign requested address" Is Returned When I Access Redis Using connect?

Symptom

Error message "Cannot assign requested address" is returned when you access Redis using **connect**.

Analysis

Applications that encounter this error typically use php-fpm and phpredis. In high-concurrency scenarios, a large number of TCP connections are in the TIME-WAIT

state. As a result, the client cannot allocate new ports and the error message will be returned.

Solutions

- Solution 1: Use **pconnect** instead of **connect**.

Using **pconnect** reduces the number of TCP connections and prevents connections from being re-established for each request, and therefore reduces latency.

When using **connect**, the code for connecting to Redis is as follows:

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

Replace **connect** with **pconnect**, and the code becomes:

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

NOTE

- Replace the connection parameters in the example with actual values. *\${Hostname}*, *\${Port}*, and *\${Inst_Password}* are the connection address, port number, and password of the Redis instance, respectively.
 - phpredis must be v5.3.0 or later. You are advised to use this **pconnect** initialization mode to avoid NOAUTH errors during disconnection.
- Solution 2: Modify the **tcp_max_tw_buckets** parameter of the ECS where the client is located.

In this solution, the ports used by TIME-WAIT connections are reused. However, if retransmission occurs between the ECS and the backend service, the connection may fail. Therefore, the **pconnect** solution is recommended.

- a. Connect to the ECS where the client is located
- b. Run the following command to check the **ip_local_port_range** and **tcp_max_tw_buckets** parameters:

```
sysctl net.ipv4.tcp_max_tw_buckets net.ipv4.ip_local_port_range
```

Information similar to the following is displayed:

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

- c. Run the following command to set the **tcp_max_tw_buckets** parameter to a value smaller than the value of **ip_local_port_range**:

```
sysctl -w net.ipv4.tcp_max_tw_buckets=10000
```

Generally, solution 1 is recommended. In special scenarios (for example, the service code involves too many components and is difficult to change), solution 2 can be used to meet high concurrency requirements.

4.16 Connection Pool Selection and Recommended Jedis Parameter Settings

Advantages of the Jedis Connection Pool

The comparison between Lettuce and Jedis is as follows:

- Lettuce
 - Lettuce does not perform connection keepalive detection. If an abnormal connection exists in the connection pool, an error is reported when requests time out.
 - Lettuce does not implement connection pool validation such as **testOnBorrow**. As a result, connections cannot be validated before being used.
- Jedis
 - Jedis implements connection pool validation using **testOnBorrow**, **testWhileIdle**, and **testOnReturn**.
If **testOnBorrow** is enabled, connection validation is performed when connections are being borrowed, which has the highest reliability but affects the performance (detection is performed before each Redis request).
 - **testWhileIdle** can be used to detect idle connections. If the threshold is set properly, abnormal connections in the connection pool can be removed in time to prevent service errors caused by abnormal connections.
 - If a connection becomes abnormal before the idle connection check, the service that uses the connection may report an error. You can specify the **timeBetweenEvictionRunsMillis** parameter to control the check interval.

Therefore, Jedis has better exception handling and detection capabilities and is more reliable than Lettuce in scenarios where there are connection exceptions and network jitters.

Recommended Jedis Connection Pool Parameter Settings

Table 4-1 Recommended Jedis connection pool parameter settings

Parameter	Description	Recommended Setting
maxTotal	Maximum number of connections	<p>Set this parameter based on the number of HTTP threads of the web container and reserved connections. Assume that the maxConnections parameter of the Tomcat Connector is set to 150 and each HTTP request may concurrently send two requests to Redis, you are advised to set this parameter to at least 400 ($150 \times 2 + 100$).</p> <p>Limit: The value of maxTotal multiplied by the number of client nodes (CCE containers or service VMs) must be less than the maximum number of connections allowed for a single DCS Redis instance.</p> <p>For example, if maxClients of a master/standby DCS Redis instance is 10,000 and maxTotal of a single client is 500, the maximum number of clients is 20.</p>
maxIdle	Maximum number of idle connections	Set this parameter to the value of maxTotal .

Parameter	Description	Recommended Setting
minIdle	Minimum number of idle connections	<p>Generally, you are advised to set this parameter to 1/X of maxTotal. For example, the recommended value is 100.</p> <p>In performance-sensitive scenarios, you can set this parameter to the value of maxIdle to prevent the impact caused by frequent connection quantity changes. For example, set this parameter to 400.</p>
maxWaitMillis	Maximum waiting time for obtaining a connection, in milliseconds	<p>The recommended maximum waiting time for obtaining a connection from the connection pool is the maximum tolerable timeout of a single service minus the timeout for command execution. For example, if the maximum tolerable HTTP timeout is 15s and the timeout of Redis requests is 10s, set this parameter to 5s.</p>
timeout	Command execution timeout, in milliseconds	<p>This parameter indicates the maximum timeout for running a Redis command. Set this parameter based on the service logic. Generally, you are advised to set this timeout to longer than 210 ms to ensure network fault tolerance. For special detection logic or environment exception detection, you can adjust this timeout to seconds.</p>

Parameter	Description	Recommended Setting
minEvictableIdleTimeMillis	Idle connection eviction time, in milliseconds. If a connection is not used for a period longer than this, it will be released.	If you do not want the system to frequently re-establish disconnected connections, set this parameter to a large value (xx minutes) or set this parameter to -1 and check idle connections periodically.
timeBetweenEvictionRunsMillis	Interval for detecting idle connections, in milliseconds	The value is estimated based on the number of idle connections in the system. For example, if this interval is set to 30s, the system detects connections every 30s. If an abnormal connection is detected within 30s, it will be removed. Set this parameter based on the number of connections. If the number of connections is too large and this interval is too short, request resources will be wasted. If there are hundreds of connections, you are advised to set this parameter to 30s. The value can be dynamically adjusted based on system requirements.
testOnBorrow	Indicates whether to check the connection validity using the ping command when borrowing connections from the resource pool. Invalid connections will be removed.	If your service is extremely sensitive to connections and the performance is acceptable, you can set this parameter to True . Generally, you are advised to set this parameter to False to enable idle connection detection.

Parameter	Description	Recommended Setting
testWhileIdle	Indicates whether to use the ping command to monitor the connection validity during idle resource monitoring. Invalid connections will be destroyed.	True
testOnReturn	Indicates whether to check the connection validity using the ping command when returning connections to the resource pool. Invalid connections will be removed.	False
maxAttempts	Number of connection retries when JedisCluster is used	Recommended value: 3–5. Default value: 5. Set this parameter based on the maximum timeout intervals of service APIs and a single request. The maximum value is 10. If the value exceeds 10, the processing time of a single request is too long, blocking other requests.

4.17 What Can I Do If a Lettuce 6.x Client Is Incompatible with My DCS Instance?

Symptom

When a Lettuce 6.x client is connected to a Proxy Cluster DCS Redis 4.x/5.x instance, the error message "NOAUTH Authentication required" is displayed.

Figure 4-3 Error message example

```
[2022-01-04 18:33:35.219] [lettuce-nioEventLoop-4-1] [DEBUG] [io.lettuce.core.AbstractRedisClient:7] ~ Connecting to Redis at 192.168.xxx.xxx:6379, initialization
java.util.concurrent.CompletionException: io.lettuce.core.RedisCommandExecutionException: NOAUTH Authentication required.
    at java.util.concurrent.CompletableFuture.encodeThrowable(CompletableFuture.java:292)
    at java.util.concurrent.CompletableFuture.completeThrowable(CompletableFuture.java:308)
```

Analysis

In Lettuce 6.x and later versions, the **HELLO** command of RESP3 (introduced in Redis 6.x) is used to determine the version adaptation. Instances of earlier versions

that do not support the **HELLO** command may encounter compatibility issues. For these instances, you can specify the RESP2 mode (compatible with Redis versions 4 and 5) in Lettuce.

Solution

Add the following code to use RESP2 protocol for accessing Redis:

```
package com.chinaroad.parking.config;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.protocol.ProtocolVersion;
import org.springframework.boot.autoconfigure.data.redis.LettuceClientConfigurationBuilderCustomizer;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;

@Configuration
public class SpringConfig implements LettuceClientConfigurationBuilderCustomizer {

    @Override
    public void customize(LettuceClientConfiguration.LettuceClientConfigurationBuilder
clientConfigurationBuilder) {
        // manually specifying RESP2
        clientConfigurationBuilder.clientOptions(ClientOptions.builder()
            .protocolVersion(ProtocolVersion.RESP2)
            .build());
    }
}
```

4.18 Should I Use a Domain Name or an IP Address to Connect to a DCS Redis Instance?

- Single-node, read/write splitting, and Proxy Cluster:
Each instance has only one IP address and one domain name address. The addresses remain unchanged before and after master/standby switchover. You can use either address to connect to the instance.
- Master/Standby (basic edition):
Each instance has one IP address and two domain name addresses. One of the domain name addresses is used only for processing read requests. The addresses remain unchanged after master/standby switchover. You can use any address to connect to the instance.

When you use a domain name address, distinguish between read and write requests. If you use **Connection Address** or **IP Address**, functions are not affected. If you use **Read-only Address**, only read requests are processed. You are advised to use read/write splitting instances if you have read/write splitting requirements.
- Redis 6.0 (professional edition)
Use the domain name for connection. There may be multiple IP addresses or they may change.
- Redis Cluster:
A Redis Cluster instance has multiple pairs of master and replica IP addresses and one domain name address. You can use any address to connect to the instance.

The connected node sends requests to the correct node. All nodes in the cluster can receive requests. **Configure multiple or all IP addresses** to prevent single points of failure.

 **NOTE**

- Domain names cannot be resolved across regions. If the client and the DCS Redis instance are not in the same region, the instance cannot be accessed using its domain name address. You can manually map the domain name to the IP address in the **hosts** file or access the instance using its IP address. For details, see [Restrictions](#).
- For details about how to connect to an instance, see [Accessing a DCS Redis Instance](#).

4.19 Is the Read-only Address of a Master/Standby Instance Connected to the Master or Standby Node?

A master/standby DCS Redis 4.0/5.0/6.0 basic instance has a **Connection Address** and a **Read-only Address**. The connection address is used to connect to the master node of the instance, and the read-only address is used to connect to the standby node of the instance.

For details, see [Architecture of Master/Standby DCS Redis 4.0/5.0/6.0 Basic Instances](#).

Figure 4-4 Instance addresses



Connection 	
Password Protected	No
Connection Address	redis-3b1cee0c-fdc8-4662-94d0-06e2e...com:6379  
Read-only Address 	redis-3b1cee0c-fdc8-4662-94d0-06e2ea...com:6379 
IP Address	10.0.0.146:6379 

By default, the client reads and writes data on the master node. Data is synchronized to the standby node. To implement read/write splitting, ensure that your client can distinguish between read and write requests. The client directs write requests to the read/write domain name and read requests to the read-only domain name.

5 Redis Usage

5.1 Can I Change the CPU Architecture?

No.

If you want to use a different CPU architecture, create another instance with the desired CPU architecture and then migrate data.

NOTE

- This function is supported by basic edition DCS Redis 4.0 instances and later, but not by professional edition DCS Redis instances.
- IP switching is supported only when both the source and target instances are Redis instances in the cloud.

Prerequisites

- The target instance is available. If you already have a DCS Redis instance, use it directly and clear the instance data before the migration. For details, see [Clearing DCS Instance Data](#).

If the target instance data is not cleared before the migration and the source and target instances contain the same key, the key in the target instance will be overwritten by the key in the source instance after the migration.

- The target Redis, source Redis, and migration task resources are in the same VPC.

NOTE

If the target and source Redis instances are not in the same VPC, ensure that the VM resources of the migration task can access these instances.

- If the source and target Redis instances are in the same region, create a VPC peering connection by referring to [VPC Peering Connection](#).
- If the source and target Redis instances are in different regions, create a cloud connection by referring to [Cloud Connect Getting Started](#).
- The target and source instances use the same port.
- IP switching can be performed only when the following conditions are met:

- IP switching depends on the data migration function. Therefore, the source and target instances must support the data migration function. For details, see [DCS data migration modes](#).
- The following table lists the supported IP switching scenarios.


Table 5-1 IP switching scenarios

Source	Target
Single-node, read/write splitting, or master/standby	Single-node, read/write splitting, master/standby, or Proxy Cluster
Proxy Cluster	Single-node, read/write splitting, master/standby, or Proxy Cluster

Precautions for IP Switching

1. Online migration will stop during the switching.
2. Instances will be read-only for one minute and disconnected for several seconds during the switching.
3. The target and source instances must use the same port.
4. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after the IP switching.
5. If the source and target instances are in different subnets, the subnet information will be updated after the switching.
6. If the source is a master/standby instance, the IP address of the standby node will not be switched. Ensure that this IP address is not used by your applications.
7. If your applications use a domain name to connect to Redis, the domain name will be used for the source instance. Select **Yes** for **Switch Domain Name**.
8. Ensure that the passwords of the source and target instances are the same. If they are different, verification will fail after the switching.
9. If a whitelist is configured for the source instance, ensure that the same whitelist is configured for the target instance before switching IP addresses.

Switching IP Addresses

- Step 1** Log in to the [DCS console](#).
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Data Migration**.
- Step 4** Click **Create Online Migration Task**.
- Step 5** Enter the task name and description.
- Step 6** Configure the VPC, subnet, and security group for the migration task.

The VPC, subnet, and security group facilitate the migration. Ensure that the migration resources can access the source and target Redis instances.

- Step 7** Configure the migration task by referring to [Configure the Online Migration Task](#). Set **Migration Type** to **Full + Incremental**.
- Step 8** On the **Online Migration** page, when the migration task status changes to **Incremental migration in progress**, choose **More > Switch IP** in the **Operation** column.
- Step 9** In the **Switch IP** dialog box, select whether to switch the domain name.

 **NOTE**


- If a domain name is used, switch it or you must modify the domain name on the client.
- If no domain name is used, the DNS of the instances will be updated.

- Step 10** Click **OK**. The IP address switching task is submitted successfully. When the status of the migration task changes to **IP switched**, the IP address switching is complete.

----End

Rolling Back IP Addresses

If you want to change the instance IP address to the original IP address, perform the following operations:

- Step 1** Log in to the [DCS console](#).
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Data Migration**.
- Step 4** On the **Online Migration** page, locate the row that contains the migration task in the **IP switched** state, choose **More > Roll Back IP**.
- Step 5** In the confirmation dialog box, click **Yes**. The IP address rollback task is submitted successfully. When the task status changes to **IP rolled back**, the rollback is complete.

----End

5.2 Can I Change the AZ for an Instance?

No.

If you want to use a different AZ, create another instance in the desired AZ and then migrate data.

 **NOTE**

- This function is supported by basic edition DCS Redis 4.0 instances and later, but not by professional edition DCS Redis instances.
- IP switching is supported only when both the source and target instances are Redis instances in the cloud.

Prerequisites

- The target instance is available. If you already have a DCS Redis instance, use it directly and clear the instance data before the migration. For details, see [Clearing DCS Instance Data](#).

If the target instance data is not cleared before the migration and the source and target instances contain the same key, the key in the target instance will be overwritten by the key in the source instance after the migration.

- The target Redis, source Redis, and migration task resources are in the same VPC.

NOTE

If the target and source Redis instances are not in the same VPC, ensure that the VM resources of the migration task can access these instances.

- If the source and target Redis instances are in the same region, create a VPC peering connection by referring to [VPC Peering Connection](#).
- If the source and target Redis instances are in different regions, create a cloud connection by referring to [Cloud Connect Getting Started](#).
- The target and source instances use the same port.
- IP switching can be performed only when the following conditions are met:
 - IP switching depends on the data migration function. Therefore, the source and target instances must support the data migration function. For details, see [DCS data migration modes](#).
 - The following table lists the supported IP switching scenarios.

Table 5-2 IP switching scenarios

Source	Target
Single-node, read/write splitting, or master/standby	Single-node, read/write splitting, master/standby, or Proxy Cluster
Proxy Cluster	Single-node, read/write splitting, master/standby, or Proxy Cluster


Precautions for IP Switching

1. Online migration will stop during the switching.
2. Instances will be read-only for one minute and disconnected for several seconds during the switching.
3. The target and source instances must use the same port.
4. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after the IP switching.
5. If the source and target instances are in different subnets, the subnet information will be updated after the switching.
6. If the source is a master/standby instance, the IP address of the standby node will not be switched. Ensure that this IP address is not used by your applications.

7. If your applications use a domain name to connect to Redis, the domain name will be used for the source instance. Select **Yes** for **Switch Domain Name**.
8. Ensure that the passwords of the source and target instances are the same. If they are different, verification will fail after the switching.
9. If a whitelist is configured for the source instance, ensure that the same whitelist is configured for the target instance before switching IP addresses.

Switching IP Addresses

Step 1 Log in to the [DCS console](#).

Step 2 Click  in the upper left corner of the management console and select the region where your instance is located.

Step 3 In the navigation pane, choose **Data Migration**.

Step 4 Click **Create Online Migration Task**.

Step 5 Enter the task name and description.

Step 6 Configure the VPC, subnet, and security group for the migration task.

The VPC, subnet, and security group facilitate the migration. Ensure that the migration resources can access the source and target Redis instances.

Step 7 Configure the migration task by referring to [Configure the Online Migration Task](#). Set **Migration Type** to **Full + Incremental**.

Step 8 On the **Online Migration** page, when the migration task status changes to **Incremental migration in progress**, choose **More > Switch IP** in the **Operation** column.

Step 9 In the **Switch IP** dialog box, select whether to switch the domain name.

NOTE

- If a domain name is used, switch it or you must modify the domain name on the client.
- If no domain name is used, the DNS of the instances will be updated.


Step 10 Click **OK**. The IP address switching task is submitted successfully. When the status of the migration task changes to **IP switched**, the IP address switching is complete.

----End

Rolling Back IP Addresses

If you want to change the instance IP address to the original IP address, perform the following operations:

Step 1 Log in to the [DCS console](#).

Step 2 Click  in the upper left corner of the management console and select the region where your instance is located.

- Step 3** In the navigation pane, choose **Data Migration**.
- Step 4** On the **Online Migration** page, locate the row that contains the migration task in the **IP switched** state, choose **More > Roll Back IP**.
- Step 5** In the confirmation dialog box, click **Yes**. The IP address rollback task is submitted successfully. When the task status changes to **IP rolled back**, the rollback is complete.
- End

5.3 Can I Change the VPC and Subnet for a DCS Redis Instance?

No. Once an instance is created, its VPC and subnet cannot be changed. If you want to use a different set of VPC and subnet, create a same instance and specify a desired set of VPC and subnet. After the new instance is created, you can migrate data from the old instance to the new instance by following the [data migration instructions](#).

5.4 Can I Customize or Change the Port for Accessing a DCS Instance?

You cannot customize or change the port for accessing a DCS Redis 3.0, 6.0 professional, or Memcached instance. You can customize and change the port for accessing a DCS Redis 4.0, 5.0, or 6.0 basic instance.

- Redis 3.0
Intra-VPC access: port 6379; public access without SSL: port 6379; public access with SSL: port 36379.
- Memcached
Use port 11211 for intra-VPC access. Public access is not supported.
- Redis 4.0/5.0/6.0
You can specify a port (ranging from 1 to 65535) or use the default port (6379) for accessing a DCS Redis 4.0/5.0/6.0 basic edition instance. If no port is specified, the default port will be used.
For Redis 6.0 professional edition, you cannot customize a port. The default port 6379 will be used.
Public access is not supported by DCS Redis 4.0/5.0/6.0 instances.


If the instance and the client are in different security groups, you must configure access rules for the security groups, allowing access through the specified port. For details, see [How Do I Configure a Security Group?](#)

Customizing a Port

When creating a DCS Redis 4.0, 5.0, or 6.0 basic edition instance, you can enter a port number for **IP Address**. If you do not specify a port, the default port 6379 is used.

Changing the Port

After a DCS Redis 4.0, 5.0, or 6.0 basic instance is created, you can change its port.

1. In the navigation pane of the DCS console, choose **Cache Manager**.
2. Click a DCS Redis instance.
3. In the **Connection** area, click  next to **Connection Address**.

NOTICE

After the port is changed, all connections to the Redis instance are interrupted, and services are connected to the new port.

5.5 Can I Modify the Connection Addresses for Accessing a DCS Instance?

After a DCS instance is created, its IP address and domain name for intra-VPC access cannot be modified. If public access has been enabled for the instance, the elastic IP address (EIP) bound to the instance can be modified.

To use a different IP address, you must create a new instance and manually specify an IP address. After the instance is created, migrate the data from the old instance to the new instance.

5.6 Why Do I Fail to Delete an Instance?

Possible causes and solutions:

- The instance is not in the **Running** state.
Only instances in the **Running** state can be deleted.
- Check whether the instance fails to be created.
To delete instances that failed to be created, click the number next to **Instance Creation Failures** on the DCS console.

5.7 Why Does It Take a Long Time to Start a Cluster DCS Instance?

Possible cause: When a cluster instance is started, status and data are synchronized between the nodes of the instance. If a large amount of data is continuously written into the instance before the synchronization is complete, the synchronization will be prolonged and the instance remains in the **Starting** state. After the synchronization is complete, the instance enters the **Running** state.

Solution: Start writing data to an instance only after the instance has been started.

5.8 What Should I Do If an Error Occurs in redis_exporter?

Start `redis_exporter` using the CLI. Based on the output, check for errors and troubleshoot accordingly.

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

5.9 What Is Reserved Memory? How Do I Configure Reserved Memory?

Reserved Memory

Reserved memory is part of the memory that is not used for storing data, but for data persistence, master/standby synchronization, and backup.

Parameter **reserved-memory-percent** is used for configuring reserved memory.

NOTICE

In the monitoring data, the memory usage does not include the usage of reserved memory.

Only the following instances must have reserved memory:

- Single-node DCS Redis 3.0 instances
- Master/standby DCS Redis 3.0 instances
- Single-node DCS Memcached instances
- Master/standby DCS Memcached instances

If reserved memory is insufficient because data occupies too much memory, the following problems may occur:

- Operations on the DCS instance become slow. (The system enables swap, deteriorating the performance.)
- Data cannot be backed up.
- Data cannot be synchronized between the master and standby nodes in time.
- Instance specifications cannot be changed.
- The process may restart.

Procedure for Configuring Reserved Memory

Change the value of **reserved-memory-percent** by referring to [Modifying Configuration Parameters of an Instance](#).

NOTE

- Set the parameter to at least **30**. For instances created in or after 2021, the default value is **30**.
- The percentage takes the maximum available memory, rather than the total memory, as the whole. The available memory is listed in the **Available Memory** column in [DCS Instance Specifications](#).

5.10 Why Is Available Memory Smaller Than Instance Cache Size?

DCS Redis 3.0 and Memcached instances are deployed on VMs, so a small amount of memory is reserved for system overheads. This problem will not occur on other instance versions.

5.11 Why Is Redisson Distributed Lock Not Supported by DCS Proxy Cluster Redis 3.0 Instances?

Redisson implements lock acquisition and unlocking in the following process:

1. Redisson lock acquisition and unlocking are implemented by running Lua scripts.
2. During lock acquisition, the **EXISTS**, **HSET**, **PEXPIRE**, **HEXISTS**, **HINCRBY**, **PEXPIRE**, and **PTTL** commands must be executed in the Lua script.
3. During unlocking, the **EXISTS**, **PUBLISH**, **HEXISTS**, **PEXPIRE**, and **DEL** commands must be executed in the Lua script.

In a proxy-based cluster, the proxy processes **PUBLISH** and **SUBSCRIBE** commands and forwards requests to the Redis server. The **PUBLISH** command cannot be executed in the Lua script.


As a result, Proxy Cluster DCS Redis 3.0 instances do not support Redisson distributed locks. **To use Redisson, resort to Redis 4.0 or 5.0 instead.**

5.12 Does DCS for Redis Provide Backend Management Software?

No. To query Redis configurations and usage information, use `redis-cli`. If you want to monitor DCS Redis instance metrics, go to the Cloud Eye console or perform the following operations.

Procedure

- Step 1** Log in to the [DCS console](#).

- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the desired instance.
- Step 5** Choose **Performance Monitoring**. All monitoring metrics of the instance are displayed.

 **NOTE**

You can also click **View Metric** in the **Operation** column on the **Cache Manager** page. You will be redirected to the Cloud Eye console. The metrics displayed on the Cloud Eye console are the same as those displayed on the **Performance Monitoring** page of the DCS console.

----End

5.13 Can I Recover Deleted Data of a DCS Instance?

If you have backed up the DCS instance, you can restore its data from the backup. However, the restoration will overwrite the data written in before the restoration.

You can restore backup data to a master/standby, cluster, or read/write splitting instance through **Backups & Restorations** on the DCS console. For details, see [Restoring a DCS Instance](#).

If a DCS instance is deleted, the instance data and its backup will also be deleted. Before deleting an instance, you can download the backup files of the instance for permanent local storage and can also migrate them to a new instance if you need to restore the data. For details about how to download the backup data, see [How Do I Export DCS Redis Instance Data?](#)

5.14 Why Is Available Memory of Unused DCS Instances Less Than Total Memory and Why Is Memory Usage of Unused DCS Instances Greater Than Zero?

For DCS Redis 3.0 instances and Memcached instances, the available memory is less than the total memory because some memory is reserved for system overhead and data persistence (supported by master/standby instances). DCS instances use a certain amount of memory for Redis-server buffers and internal data structures. This is why memory usage of unused DCS instances is greater than zero. This problem will not occur on other instance versions.

5.15 How Do I Check Redis Memory Usage?

Currently, DCS for Redis provides the following memory-related metrics: For details about how to check monitoring metrics, see [Viewing Metrics](#).

Table 5-3 DCS Redis instance metrics

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Raw Data)
memory_usage	Memory Usage	Memory consumed by the monitored object Unit: %	0–100%	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory	Used Memory	Number of bytes used by the Redis server Unit: KB, MB, or byte (configurable on the console)	≥ 0	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory_dataset	Used Memory Dataset	Dataset memory that the Redis server has used Unit: KB, MB, or byte (configurable on the console)	≥ 0	Monitored object: DCS Redis instance Only Redis 4.0 and later Dimension: dcs_instance_id	1 minute
used_memory_dataset_perc	Used Memory Dataset Ratio	Percentage of data memory that Redis has used to the total used memory Unit: %	0–100%	Monitored object: DCS Redis instance Only Redis 4.0 and later Dimension: dcs_instance_id	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Raw Data)
used_memory_rss	Used Memory RSS	Resident set size (RSS) memory that the Redis server has used, which is the memory that actually resides in the memory, including all stack and heap memory but not swapped-out memory Unit: KB, MB, or byte (configurable on the console)	≥ 0	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute
memory_fragmentation_ratio	Memory Fragmentation Ratio	Current memory fragmentation, which is the ratio between used_memory_rss/used_memory .	≥ 0	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory_peak	Used Memory Peak	Peak memory consumed by Redis since the Redis server last started Unit: KB, MB, or byte (configurable on the console)	≥ 0	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory_lua	Used Memory Lua	Number of bytes used by the Lua engine Unit: KB, MB, or byte (configurable on the console)	≥ 0	Monitored object: DCS Redis instance Dimension: dcs_instance_id	1 minute

5.16 Why Is the Capacity or Performance of a Shard of a Redis Cluster Instance Overloaded When That of the Instance Is Still Below the Bottleneck?

Redis Cluster uses a special data sharding method. **Every key is part of a hash slot, which is held by a node in the cluster.** To compute what is the hash slot of a given key:

1. Take the CRC16 of the key modulo 16384.
2. Based on the mapping between hash slots and shards, connections are redirected to the right node for data read and write operations.

Therefore, keys are not evenly distributed to each shard of an instance. If a shard contains a big key or a hot key, the capacity or performance of the shard will be overloaded, but the load on other shards is still low. As a result, the capacity or performance bottleneck of the entire instance is not reached.

5.17 Why Does an OOM Error Occur During a Redis Connection?

Symptom

"Error in execution; nested exception is io.lettuce.core.RedisCommandExecutionException: OOM command not allowed when used memory > 'maxmemory'" is returned during a Redis connection.

Fault Locating

An out-of-memory (OOM) error indicates that the maximum memory is exceeded. In the error information, the **maxmemory** parameter indicates the maximum memory configured on the Redis server.

If the memory usage of the Redis instance is less than 100%, the memory of the node where data is written may have reached the maximum limit. Connect to each node in the cluster by running **redis-cli -h <redis_ip> -p 6379 -a <redis_password> -c --bigkeys**. When connecting to a replica node, run the **READONLY** command before running the **bigkeys** command.

5.18 What Clients Can I Use for Redis Cluster in Different Programming Languages?

The following table compares Redis Cluster and Proxy Cluster in DCS.

Table 5-4 Comparing Redis Cluster and Proxy Cluster

Item	Redis Cluster	Proxy Cluster
Redis compatibility	High	Medium
Client compatibility	Medium (The cluster mode must be enabled on the client.)	High
Costs	High	Medium
Latency	Low	Medium
Read/write splitting	Native support (client SDK configuration)	Implemented by using proxies
Performance	High	Medium

Redis Cluster does not use proxies, and therefore delivers lower latency and higher performance. However, Redis Cluster instances are based on the open-source Redis Cluster protocol, so their client compatibility is poorer than that of Proxy Cluster instances.

The following table lists clients that can be used for Redis Cluster.

Table 5-5 Clients that can be used for Redis Cluster

Language	Client	Reference Document
Java	Jedis	https://github.com/xetorthio/jedis#jedis-cluster
Java	Lettuce	https://github.com/lettuce-io/lettuce-core/wiki/Redis-Cluster
PHP	php redis	https://github.com/phpredis/phpredis#readme
Go	Go Redis	Redis Cluster: https://pkg.go.dev/github.com/go-redis/redis/v8#NewClusterClient Proxy Cluster, single-node, or master/standby: https://pkg.go.dev/github.com/go-redis/redis/v8#NewClient

Language	Client	Reference Document
Python	redis-py-cluster	https://github.com/Grokzen/redis-py-cluster#usage-example
C	hireis-vip	https://github.com/vipshop/hireis-vip?_ga=2.64990636.268662337.1603553558-977760105.1588733325
C++	redis-plus-plus	https://github.com/sewnew/redis-plus-plus?_ga=2.64990636.268662337.1603553558-977760105.1588733325#redis-cluster
Node.js	node-redis io-redis	https://github.com/NodeRedis/node-redis https://github.com/luin/iodis

To view all Redis clients, see <https://redis.io/clients>.

5.19 Why Do I Need to Configure Timeout for Redis Cluster?

If timeout is not configured, connections will fail.

When you connect to a Redis Cluster instance using Spring Boot and Lettuce, connect to all cluster nodes, including faulty replicas.

- If timeout is not configured, minute-level blocking (120s in earlier Lettuce versions and 60s in the new version) may occur when there is a faulty replica, as shown in the following figure.

```

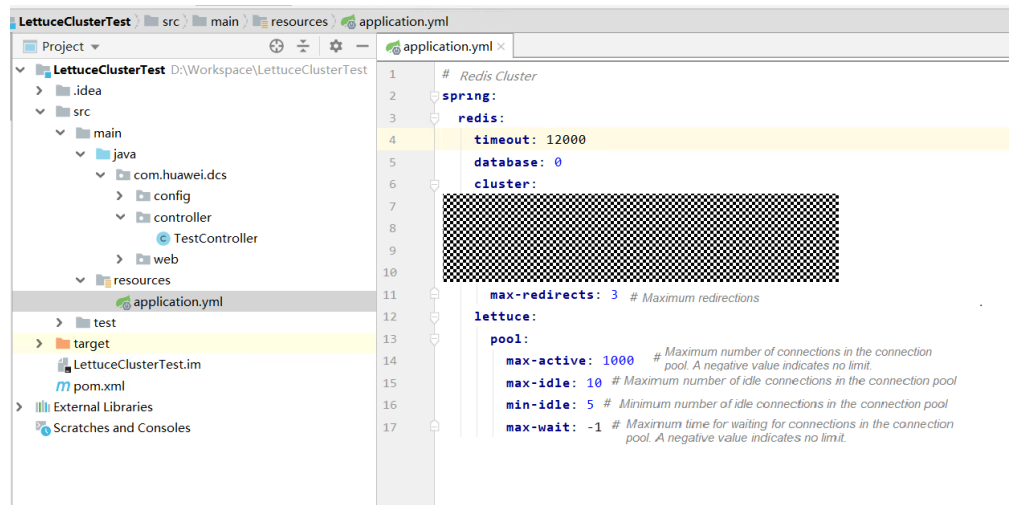
1 # Redis Cluster
2 spring:
3   redis:
4     database: 0
5     cluster:
6     [Redacted]
7     [Redacted]
8     [Redacted]
9     nodes: [Redacted]
10    max-redirects: 3 # Maximum redirections
11 lettuce:
12   pool:
13     max-active: 1000 # Maximum number of connections in the connection
14     pool. A negative value indicates no limit.
15     max-idle: 10 # Maximum number of idle connections in the connection pool
16     min-idle: 5 # Minimum number of idle connections in the connection pool
17     max-wait: -1 # Maximum time for waiting for connections in the connection
18     pool. A negative value indicates no limit.

```

The end-to-end service access time may reach the maximum timeout, as shown in the following figure.

```
[root@ecs-a776 ~]# time curl -X get http://177.0.0.1:8080/test/evalsha
false
real    2m0.632s
user    0m0.003s
sys     0m0.004s
```

- After the **timeout** parameter is configured on the client, the timeout on the replica will be greatly shortened. You can adjust the timeout based on the service requirements. Assume that the configuration is as follows.



The following figure shows the end-to-end service access time after the configuration is complete.

```
[root@ecs-a776 ~]# time curl -X get http://177.0.0.1:8080/test/evalsha
false
real    0m12.627s
user    0m0.000s
sys     0m0.004s
```

If the **timeout** parameter is not configured and there is a faulty node, client connections will be blocked.

Configure the timeout based on what the service can tolerate. For example, if you need to request Redis twice in an HTTP request and the maximum timeout of an HTTP request is 10s, it is recommended that you set the timeout in Redis to 5s. This prevents service interruption if faults occur due to a long timeout duration or no timeout duration.

5.20 Why Am I Seeing a Timeout Error When Reading Data from Redis?

Symptom

When you read data from Redis, timeout error "redis server response timeout (3000ms) occurred after 3 retry attempts" is returned.

```
021-05-28 20:11:13.479 [ERROR] [TR:13a974b3121094a41d91b8d94ea25a31f] [http-nio-8110-exec-1] [get redis exception: [RedisRepository.java:132 org.springframework.dao.QueryTi
outException: Redis server response timeout (3000 ms) occurred after 3 retry attempts. Increase nettyThreads and/or timeout settings. Try to define pingConnectionInterval sett
ing. Command: (GET), params: [184, 69, 78, 65, 78, 84, 95, 65, 76, 76, ...]]; channel: [id: 0x0a600f4-tty72-2000000-40-46690-~rredis-master.enterpriseadmin.middleware.local/192
.168.1.100:6379]; nested exception is org.redisson.client.RedisResponseTimeoutException: Redis server response timeout (3000 ms) occurred after 3 retry attempts. Increase nettyT
hreads and/or timeout settings. Try to define pingConnectionInterval setting. Command: (GET), params: [184, 69, 78, 65, 78, 84, 95, 65, 76, 76, ...]]; channel: [id: 0x0a600f4-
L1/17] [0:46690-~rredis-master.enterpriseadmin.middleware.local/192.168.1.100:6379]
at org.redisson.spring.data.connection.RedissonExceptionConverter.convert(RedissonExceptionConverter.java:48) ~[redisson-spring-data-23-3.13.5.jar:3.13.5]
at org.redisson.spring.data.connection.RedissonExceptionConverter.convert(RedissonExceptionConverter.java:35) ~[redisson-spring-data-23-3.13.5.jar:3.13.5]
at org.springframework.data.redis.PassThroughExceptionTranslationStrategy.translate(PassThroughExceptionTranslationStrategy.java:44) ~[spring-data-redis-2.2.10.RELEASE-
jar:2.2.10.RELEASE]
at org.springframework.data.redis.connection.RedissonConnection.translate(RedissonConnection.java:223) ~[redisson-spring-data-23-3.13.5.jar:3.13.5]
```

Troubleshooting

1. Increase the timeout based on the error information.
2. Check whether the error is returned after an operation is performed on a big key. It is recommended that the key size be no greater than 10 KB.
Redis limits the size of each String value to 512 MB. In actual development, keep the size within 10 KB. Otherwise, the CPU and NIC will be heavily loaded.
Keep the number of Hashes, Lists, Sets, or Zsets within 5000.
Theoretically, the number of elements in each HashSet should be fewer than $2^{\wedge}32$.
3. Increase the value of parameter **PingConnectionInterval** based on the error information.

5.21 Explaining and Using Hash Tags

Hash Tag Design

Multi-key operations, such as those using the **MSET** command or Lua scripts, are atomic. All specified keys are executed at the same time. However, in a cluster, each key is hashed to a given shard, and multi-key operations are no longer atomic. The keys may be allocated to different slots. As a result, some keys are updated, while others are not. If there is a hash tag, the cluster determines which slot to allocate a key based on the hash tag. Keys with the same hash tag are allocated to the same slot.

Using Hash Tags

Only the content between the first left brace ({) and the following first right brace (}) is hashed.

For example:

- In keys **{user1000}.following** and **{user1000}.followers**, there is only one pair of braces. **user1000** will be hashed.
- In key **foo{bar}**, there is no content between the first { and the first }. The whole key **foo{bar}** will be hashed as usual.
- In key **foo{bar}zap**, **bar** (the content between the first { and the first }) is hashed.
- In key **foo{bar}zap**, **bar** is hashed because it is between the first pair of { and }.

Hash Tag Example

When the following operation is performed:

```
EVAL "redis.call('set',KEYS[1],ARGV[1]) redis.call('set',KEYS[2],ARGV[2])" 2 key1 key2 value1 value2
```

The following error is displayed:

```
ERR 'key1' and 'key2' not in the same slot
```

You can use a hash tag to solve this issue:

```
EVAL "redis.call('set',KEYS[1],ARGV[1]) redis.call('set',KEYS[2],ARGV[2])" 2 {user}key1 {user}key2 value1 value2
```

5.22 Why Does a Key Disappear in Redis?

Normally, Redis keys do not disappear. If a key is missing, it may have expired, been evicted, or been deleted.

Perform the following checks one by one:

1. Check whether the key has expired.
2. View the monitoring information and check whether eviction was triggered.
3. Run the **INFO** command on the server side to check whether the key has been deleted.

5.23 Will Cached Data Be Retained After an Instance Is Restarted?

After a single-node DCS instance is restarted, data in the instance is deleted.

Master/Standby, read/write splitting, and cluster instances (except single-replica clusters) support AOF persistence by default. Data is retained after these instances are restarted.

If AOF persistence is disabled (**appendonly** is set to **no**), data is deleted after the instances are restarted.

5.24 How Do I Know Whether an Instance Is Single-DB or Multi-DB?

Single-node, read/write splitting, and master/standby: multi-DB (256 DBs, numbered from 0 to 255)

Proxy Cluster: single-DB by default. Multi-DB can be enabled. For details, see [What Are the Constraints on Implementing Multiple Databases on a Proxy Cluster Instance?](#)

Redis Cluster: single-DB. Multi-DB is not supported.

You can connect to a DCS Redis 4.0 or later instance on the console to check whether it is multi-DB.

Figure 5-1 Connecting to Redis

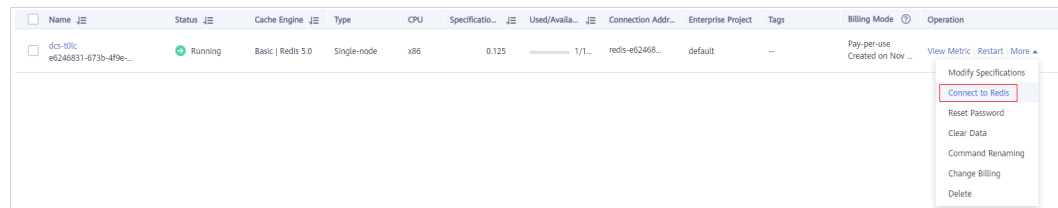
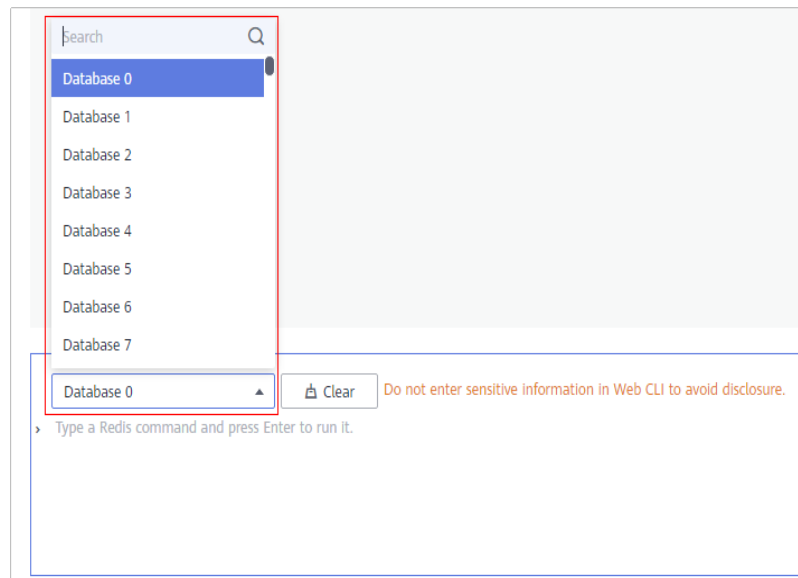


Figure 5-2 Viewing databases



5.25 What Are the Constraints on Implementing Multi-DB on a Proxy Cluster Instance?

Note the following constraints when you consider implementing multi-DB:

- **Usage constraints:**
 - a. The **SWAPDB** command does not support multi-DB.
 - b. The **INFO KEYSpace** command does not return data of multi-DB.
 - c. To query the total number of keys in each database, use the customized **dbstats** command. CPU usage will surge on the node executing this command.
 - d. LUA scripts do not support multi-DB.
 - e. The **RANDOMKEY** command does not support multi-DB.
 - f. The **SELECT** command cannot be embedded in transactions.
 - g. **PUBLISH** cannot be used in Lua scripts.
 - h. The database number ranges from 0 to 255.
- **Performance constraints**
 - a. The **FLUSHDB** command deletes keys one by one, which takes a long time and is slower than the open-source native implementation. The

- execution speed of the **FLUSHDB** command is the same as that of the **SCAN** command (which should be tested by the customer).
 - b. The **DBSIZE** command is time-consuming. Do not use it in the code.
 - c. If multi-DB is used, the performance of the **KEYS** and **SCAN** commands deteriorates by up to 50%.
 - **Other constraints:**

The backend storage rewrites keys based on certain rules. Keys in the exported RDB file are not the original keys. However, the access through the Redis protocol is not affected.

Procedure for Enabling or Disabling Multi-DB

By default, a Proxy Cluster instance is not enabled with multi-DB, but you can do as follows to enable multi-DB.

Step 1 Log in to the DCS console.

Step 2 Connect to the instance and run the **FLUSHALL** command to clear the instance data.

NOTE

Before enabling or disabling multi-DB, ensure that all instance data is cleared and no data is being written.

Step 3 On the **Cache Manager** page of the DCS console, click the desired DCS instance.

Step 4 Choose **Instance Configuration > Parameters**.

Step 5 To enable multi-DB, click **Modify** in the row that contains parameter **multi-db**, and change its value to **yes**.

To disable multi-DB, change the value to **no**.

Step 6 Click **Save** and then confirm the modification. The instance does not need to be restarted.

maxmemory-policy ⓘ	volatile-lru	volatile-lru,allkeys-lru,volatile-lfu,allkeys-lfu,volatile-random,allkeys-random,volatile-t...	volatile-lru	Modify
multi-db ⓘ	no	no,yes	no	Modify
multi-db-keys-scan-enabled ⓘ	no	no,yes	no	Modify

----End


5.26 How Do I Create a Multi-DB Proxy Cluster Instance?

When you create a Proxy Cluster instance, there is only one database by default. This section describes how to create a Proxy Cluster instance with multiple databases.

NOTE

Before getting started, learn about [the constraints on implementing multi-DB](#).

Step 1 Log in to the [DCS console](#).

Step 2 Click  in the upper left corner to select a region.

Step 3 In the navigation pane, choose **Parameter Templates**.

Step 4 In the row that contains the template with the desired cache engine version and instance type (Proxy Cluster), click **Customize**.

Step 5 Set **multi-db** to **yes**.

Step 6 Enter a new template name and click **OK**. The custom template is created successfully.

Step 7 In the navigation pane, choose **Cache Manager**. Then click Buy DCS Instance to create a Proxy Cluster instance.

Set **Parameter Configuration** to **Use custom template** and select the custom template created in the preceding step.



After the instance is created, connect to it to check whether it has multiple databases.

----End

6 Instance Scaling and Upgrade

6.1 Can I Upgrade Version for a DCS Redis Instance, for Example, from Redis 4.0 to Redis 5.0?

No. Different Redis versions use different underlying architectures. The Redis version used by a DCS instance cannot be changed once the instance is created.

If your service requires the features of higher Redis versions, create a new DCS Redis instance of a higher version and then migrate data from the original instance to the new one. For details on how to migrate data, see the [Data Migration Guide](#).

6.2 How Do I Upgrade a Single-Node or Master/Standby Redis 3.0 Instance?

Background

Redis has not updated Redis 3.0 since the release of a minor version on May 19, 2019. Huawei Cloud DCS also announced the discontinuation of DCS for Redis 3.0 in March 2021.

You are advised to upgrade your Redis 3.0 to a later version as soon as possible. DCS for Redis 4.0/5.0/6.0 are compatible with Redis 3.0.

Follow the instructions below to upgrade a single-node or master/standby DCS Redis 3.0 instance by migrating data.

 **NOTE**

1. The instructions apply only to upgrade from a single-node or master/standby DCS Redis 3.0 instance to a single-node or master/standby DCS Redis 4.0/5.0/6.0 instance.
2. DCS Redis 3.0 instances support public network access, while DCS Redis 4.0/5.0/6.0 instances do not. If your services depend on public network access, do not perform the upgrade.
3. Upgrading the Redis version through data migration may have the following impacts on services:
 - If you choose the full + incremental migration type, and the source and target instances have the same number of cached keys or the offset is small, services must stop writing data during incremental migration.
 - After data synchronization is complete, switch the domain name or IP address and the password (if the target and source instances have different passwords) for accessing Redis. Services must be stopped during the switch.
 - After switching the instance connection address and password, check whether services are running properly and whether key performance metrics are normal.
4. Upgrade instances during off-peak hours.

Prerequisites

- You have created a DCS Redis instance of a later version. This instance must be in the same VPC, of the same instance type, preferably configured with the same password as the source Redis 3.0 instance, and have specifications greater than or equal to the source instance.
For example, to upgrade a 16 GB master/standby DCS Redis 3.0 instance to DCS Redis 5.0, prepare a master/standby DCS Redis 5.0 instance with at least 16 GB memory.
- You have manually backed up data of the source DCS Redis 3.0 instance.

Migrating Data

Step 1 Log in to the DCS console.

Step 2 In the navigation pane, choose **Data Migration**. The migration task list is displayed.

Step 3 In the upper right corner, click **Create Online Migration Task**.

Step 4 Enter the task name and description.

Step 5 Configure the VPC, subnet, and security group for the migration task.

Select the VPC where the source and target Redis instances are, so they can be connected with the task.

The migration task uses a tenant IP address (**Migration ECS** displayed on the **Basic Information** page of the task). If the target Redis has a whitelist, this IP address must be added to the whitelist.

Step 6 After the migration task is created, click **Configure** in the **Operation** column of the task on the **Online Migration** tab page to configure the source Redis and target Redis.

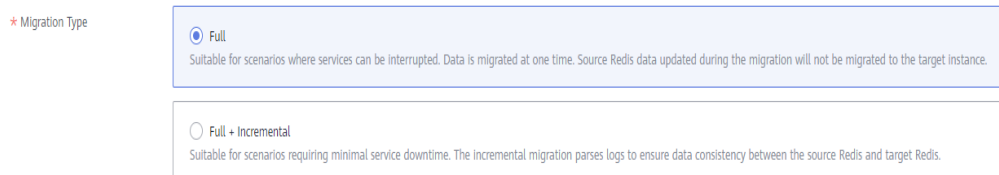
Step 7 Select a migration type.

Supported migration types are **Full** and **Full + Incremental**, which are described in [Table 6-1](#).

Table 6-1 Migration type description

Migration Type	Description
Full	Suitable for scenarios where services can be interrupted. Data is migrated at one time. Source instance data updated during the migration will not be migrated to the target instance.
Full + incremental	Suitable for scenarios requiring minimal service downtime. The incremental migration parses logs to ensure data consistency between the source and target instances. Once the migration starts, it remains Migrating until you click Stop in the Operation column. After the migration is stopped, data in the source instance will not be lost, but data will not be written to the target instance. When the transmission network is stable, the delay of incremental migration is within seconds. The actual delay depends on the transmission quality of the network link.

Figure 6-1 Selecting the migration type



Step 8 If **Migration Type** is set to **Full + Incremental**, you can specify a bandwidth limit.

The data synchronization rate can be kept around the bandwidth limit.

Step 9 Specify **Auto-Reconnect**. If this option is enabled, automatic reconnections will be performed indefinitely in the case of a network exception.

Full synchronization will be triggered and requires more bandwidth if incremental synchronization becomes unavailable. Exercise caution when enabling this option.

Step 10 For **Source Redis Instance**, select the DCS Redis 3.0 instance to be upgraded.

If the source DCS Redis 3.0 instance is password-protected, click **Test Connection** next to **Source Redis Instance Password** to check whether the instance password is correct and whether the network is connected.

Step 11 For **Target Redis Instance**, select the new higher version DCS Redis instance that you have prepared.

If the target instance is password-protected, click **Test Connection** next to **Target Redis Instance Password** to check whether the instance password is correct and whether the network is connected.

Step 12 Click **Next**.

Step 13 Confirm the migration task details and click **Submit**.

Go back to the data migration task list. After the migration is successful, the task status changes to **Successful**.

 **NOTE**

- Once incremental migration starts, it remains **Migrating**.
- To manually stop migration, click **Stop**.
- After data migration, duplicate keys will be overwritten.

If the migration fails, click the migration task and check the log on the **Migration Logs** page.

----End

Verifying the Migration

After the migration is complete, use redis-cli to connect the source and target Redis instances to check data integrity.

1. Connect to the source Redis and the target Redis.
2. Run the **info keyspace** command to check the values of **keys** and **expires**.

```
192.168.1.217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.168.1.217:6379>
```

3. Calculate the difference between the values of **keys** and **expires** of the source Redis and the target Redis. If the differences are the same, the data is complete and the migration is successful.

During full migration, source Redis data updated during the migration will not be migrated to the target instance.

Switching the Redis Address and Verifying Services

Step 1 Change the domain name or IP address that your service uses to connect to Redis from that of the source Redis 3.0 instance to that of the new higher version instance.

Figure 6-2 Obtaining the instance connection addresses



Step 2 If the password of the new instance is different from that of the original instance, change the password on the client.

- Step 3** Verify that service functions are normal. For example, check whether an error is reported when the client accesses Redis.
- Step 4** Check whether key performance metrics are normal, such as **Connected Clients**, **Ops per Second**, **CPU Usage**, and **Memory Usage**.

----End

6.3 Are Services Interrupted If Maintenance is Performed During the Maintenance Time Window?

O&M personnel will contact you before performing maintenance during the maintenance time window, informing you of the operations and their impacts. You do not need to worry about instance running exceptions.

6.4 Are Instances Stopped or Restarted During Specification Modification?

No. Specification modifications can take place while the instance is running and do not affect any other resources.

6.5 What DCS Instance Type Changes Are Supported?

Table 6-2 Instance type change options supported by different DCS instances

Version	Supported Type Change	Precautions
Redis 3.0	From single-node to master/standby	The instance cannot be connected for several seconds and remains read-only for about one minute.
	From master/standby to Proxy Cluster	<ol style="list-style-type: none">1. If the data of a master/standby DCS Redis 3.0 instance is stored in multiple databases, or in non-DB0 databases, the instance cannot be changed to the Proxy Cluster type. A master/standby instance can be changed to the Proxy Cluster type only if its data is stored only on DB0.2. The instance cannot be connected and remains read-only for 5 to 30 minutes.
Memcached	From single-node to master/standby	Services are interrupted for several seconds and remain read-only for about 1 minute.

Version	Supported Type Change	Precautions
Redis 4.0/5.0	From master/standby or read/write splitting to Proxy Cluster	<ol style="list-style-type: none"> 1. Before changing the instance type to Proxy Cluster, evaluate the impact on services. For details, see What Are the Constraints on Implementing Multi-DB on a Proxy Cluster Instance? and Command Restrictions. 2. Memory usage must be less than 70% of the maximum memory of the new flavor. 3. Some keys may be evicted if the current memory usage exceeds 90% of the total. 4. After the change, create alarm rules again for the instance. 5. For instances that are currently master/standby, ensure that their read-only IP address or domain name is not used by your application. 6. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after the change. 7. Modify instance specifications during off-peak hours. An instance is temporarily interrupted and remains read-only for about 1 minute during the specification change.
	From Proxy Cluster to master/standby or read/write splitting	

For details about the commands supported by different types of instances, see [Command Compatibility](#).

Any instance type changes not listed in the preceding table are not supported. To modify specifications while changing the instance type, see [IP Switching](#).

To check whether you can change the instance type of an instance, see the parameters displayed on the [Modify Specifications](#) page on the DCS console.

6.6 Are Services Interrupted During Specification Modification?

Modify instance specifications during off-peak hours.

If the modification failed in peak hours (for example, when memory or CPU usage is over 90% or write traffic surges), try again during off-peak hours.

The following table describes the impact of specification modification.

Change of the Instance Type

Table 6-3 Instance type change options supported by different DCS instances

Version	Supported Type Change	Precautions
Redis 3.0	From single-node to master/standby	The instance cannot be connected for several seconds and remains read-only for about one minute.
	From master/standby to Proxy Cluster	<ol style="list-style-type: none"> 1. If the data of a master/standby DCS Redis 3.0 instance is stored in multiple databases, or in non-DB0 databases, the instance cannot be changed to the Proxy Cluster type. A master/standby instance can be changed to the Proxy Cluster type only if its data is stored only on DB0. 2. The instance cannot be connected and remains read-only for 5 to 30 minutes.
Memcached	From single-node to master/standby	Services are interrupted for several seconds and remain read-only for about 1 minute.

Version	Supported Type Change	Precautions
Redis 4.0/5.0	From master/standby or read/write splitting to Proxy Cluster	<ol style="list-style-type: none"> 1. Before changing the instance type to Proxy Cluster, evaluate the impact on services. For details, see What Are the Constraints on Implementing Multi-DB on a Proxy Cluster Instance? and Command Restrictions. 2. Memory usage must be less than 70% of the maximum memory of the new flavor. 3. Some keys may be evicted if the current memory usage exceeds 90% of the total. 4. After the change, create alarm rules again for the instance. 5. For instances that are currently master/standby, ensure that their read-only IP address or domain name is not used by your application. 6. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after the change. 7. Modify instance specifications during off-peak hours. An instance is temporarily interrupted and remains read-only for about 1 minute during the specification change.
	From Proxy Cluster to master/standby or read/write splitting	

Any instance type changes not listed in the preceding table are not supported. To modify specifications while changing the instance type, see [IP Switching](#).

For details about the commands supported by different types of instances, see [Command Compatibility](#).

Scaling

- **Scaling options**

Table 6-4 Scaling options supported by different instances

Cache Engine	Single-Node	Master/Standby	Redis Cluster	Proxy Cluster	Read/Write Splitting
Redis 3.0	Scaling up/down	Scaling up/down	-	Scaling out	-

Cache Engine	Single-Node	Master/Standby	Redis Cluster	Proxy Cluster	Read/Write Splitting
Redis 4.0	Scaling up/down	Scaling up/down and replica quantity change	Scaling up/down, out/in, and replica quantity change	Scaling up/down and out/in	Scaling up/down and replica quantity change
Redis 5.0	Scaling up/down	Scaling up/down and replica quantity change	Scaling up/down, out/in, and replica quantity change	Scaling up/down and out/in	Scaling up/down and replica quantity change
Redis 6.0 basic edition	Scaling up/down	Scaling up/down	Scaling up/down, out/in, and replica quantity change	-	-
Redis 6.0 professional editions	-	Scaling up/down	-	-	-
Memcached	Scaling up/down	Scaling up/down	-	-	-

 **NOTE**

- If the reserved memory of a DCS Redis 3.0 or Memcached instance is insufficient, the modification may fail when the memory is used up. For details, see [Reserved Memory](#).
- Change the replica quantity and capacity separately.
- **Impact of scaling**

Table 6-5 Impact of scaling

Instance Type	Scaling Type	Impact
Single-node , read/write splitting, and master/standby	Scaling up/down	<ul style="list-style-type: none">• During scaling up, a DCS Redis 4.0/5.0/6.0 basic edition instance will be disconnected for several seconds and remain read-only for about 1 minute. During scaling down, connections will not be interrupted.• A DCS Redis 3.0 instance will be disconnected for several seconds and remain read-only for 5 to 30 minutes.• A DCS Redis professional edition instance will be disconnected for several seconds and remain read-only for about 1 minute.• For scaling up, only the memory of the instance is expanded. The CPU processing capability is not improved.• Single-node DCS instances do not support data persistence. Scaling may compromise data reliability. After scaling, check whether the data is complete and import data if required. If there is important data, use a migration tool to migrate the data to other instances for backup.• For master/standby and read/write splitting instances, backup records created before scale-down cannot be used after scale-down. If necessary, download the backup file in advance or back up the data again after scale-down.

Instance Type	Scaling Type	Impact
Proxy Cluster and Redis Cluster	Scaling up/down	<ul style="list-style-type: none"> ● Scaling out by adding shards: <ul style="list-style-type: none"> – Scaling out does not interrupt connections but will occupy CPU resources, decreasing performance by up to 20%. – If the shard quantity increases, new Redis Server nodes are added, and data is automatically balanced to the new nodes, increasing the access latency. ● Scaling in by reducing shards: <ul style="list-style-type: none"> – If the shard quantity decreases, nodes will be deleted. Before scaling in a Redis Cluster instance, ensure that the deleted nodes are not directly referenced in your application, to prevent service access exceptions. – Nodes will be deleted, and connections will be interrupted. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after scaling. ● Scaling up by shard size without changing the shard quantity: <ul style="list-style-type: none"> – Insufficient memory of the node's VM will cause the node to migrate. Service connections may stutter and the instance may become read-only during the migration. – Increasing the node capacity when the VM memory is sufficient does not affect services. <p>NOTE Scaling up by increasing the shard size is available on a whitelist basis. Contact technical support to use this feature.</p> ● Scaling down by reducing the shard size without changing the shard quantity has no impact. ● To scale down an instance, ensure that the used memory of each node is less than 70% of the maximum memory per node of the new flavor. ● The flavor changing operation may involve data migration, and the latency may increase. For a Redis Cluster instance, ensure that the client can process the MOVED and ASK commands. Otherwise, the request will fail. ● If the memory becomes full during scaling due to a large amount of data being written, scaling will fail. ● Before scaling, check for big keys through Cache Analysis. Redis has a limit on key migration. If the instance has any single key greater than 512 MB,

Instance Type	Scaling Type	Impact
		<p>scaling will fail when big key migration between nodes times out. The bigger the key, the more likely the migration will fail.</p> <ul style="list-style-type: none"> • Before scaling a Redis Cluster instance, ensure that automated cluster topology refresh is enabled. If it is disabled, you will need to restart the client after scaling. For details about how to enable automated refresh if you use Lettuce, see an example of using Lettuce to connect to a Redis Cluster instance. • Backup records created before scaling cannot be used. If necessary, download the backup file in advance or back up the data again after scaling.
Master/standby, read/write splitting, and Redis Cluster instances	Scaling out/in (replica quantity change)	<ul style="list-style-type: none"> • Before adding or removing replicas for a Redis Cluster instance, ensure that automated cluster topology refresh is enabled. If it is disabled, you will need to restart the client after scaling. For details about how to enable automated refresh if you use Lettuce, see an example of using Lettuce to connect to a Redis Cluster instance. • Deleting replicas interrupts connections. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after scaling. Adding replicas does not interrupt connections. • If the number of replicas is already the minimum supported by the instance, you can no longer delete replicas.

6.7 Why Do I Fail to Modify the Specifications for a DCS Instance?

- Check whether other tasks are running.
Specifications of a DCS instance cannot be modified if another task of the instance is still running. For example, you cannot delete or scale up an instance while it is being restarted. Likewise, you cannot delete an instance while it is being scaled up.
If the specification modification fails, try again later. If it fails again, contact technical support.
- When changing a master/standby instance to the Proxy Cluster type, check whether data exists in DBs other than DB0. Specification modification will fail if a DB other than DB0 contains data.

A master/standby instance can be changed to the Proxy Cluster type when data exists only in DB0.

6.8 How Do I Reduce the Capacity of a DCS Instance?

[Table 6-6](#) lists scaling options supported by different DCS instances.

Table 6-6 Scaling options supported by different instances

Cache Engine	Single-Node	Master/Standby	Redis Cluster	Proxy Cluster	Read/Write Splitting
Redis 3.0	Scaling up/down	Scaling up/down	-	Scaling out	-
Redis 4.0	Scaling up/down	Scaling up/down and replica quantity change	Scaling up/down, out/in, and replica quantity change	Scaling up/down and out/in	Scaling up/down and replica quantity change
Redis 5.0	Scaling up/down	Scaling up/down and replica quantity change	Scaling up/down, out/in, and replica quantity change	Scaling up/down and out/in	Scaling up/down and replica quantity change
Redis 6.0 basic edition	Scaling up/down	Scaling up/down	Scaling up/down, out/in, and replica quantity change	-	-
Redis 6.0 professional editions	-	Scaling up/down	-	-	-
Memcached	Scaling up/down	Scaling up/down	-	-	-

For details about how to change the capacity, see [Modifying Specifications](#).

If you want to use a smaller Proxy Cluster DCS Redis 3.0 instance, back up the data of the existing instance, and create a new Proxy Cluster instance with the desired capacity. Then, import the backup data to the new instance. After the data

migration is complete, delete the old instance. For details about data migration operations, see [Importing Backup Files](#).

6.9 How Do I Add Shards to a Cluster DCS Redis Instance Without Changing the Memory?

After a Proxy Cluster or Redis Cluster instance is created, you can reduce the capacity of each shard and add more shards without changing the total memory.

For example, if an 8 GB instance has 4 shards and each shard is 2 GB, you can reduce the shard size to 1 GB and increase the shard quantity to 8.

NOTE

A shard size of 1 GB cannot be changed.

Procedure


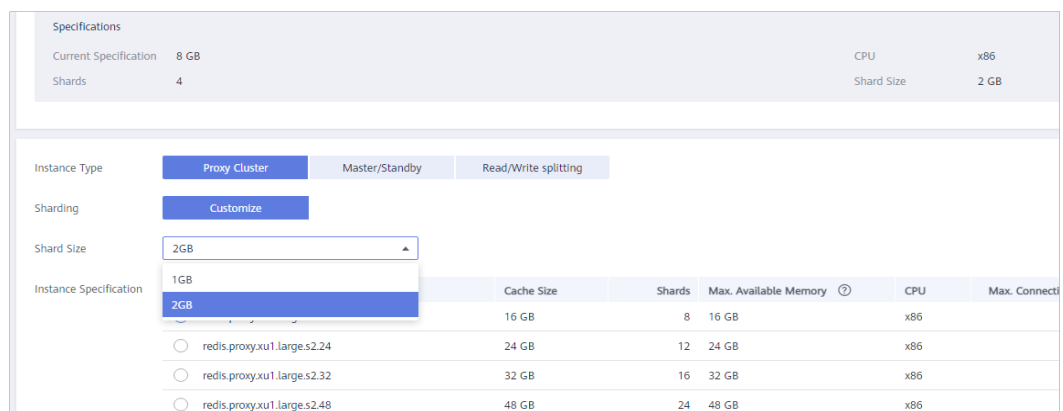
- Step 1** Log in to the [DCS console](#).
- Step 2** Click  in the upper left corner to select a region and a project.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Choose **More > Modify Specifications** in the row containing the desired DCS instance.
- Step 5** On the displayed **Modify Instance Specifications** page, specify **Shard Size** and **Instance Specification**.

Figure 6-3 Selecting a shard size



- Step 6** Click **Next**, confirm the details, and click **Submit**.

The modification takes about 5 to 30 minutes to complete. After the modification is successful, the instance status changes to **Running**.

----End

6.10 How Do I Handle an Error When I Use Lettuce to Connect to a Redis Cluster Instance After Specification Modification?

Symptom

If the shard quantity changes during specification modification of a Redis Cluster instance, some slots are migrated to new shards. The following error occurs when you use Lettuce to connect to the instance.

Figure 6-4 Error

```
org.springframework.data.redis.RedisSystemException: Redis exception; nested exception is io.lettuce.core.RedisException: io.lettuce.core.RedisException: java.lang.IllegalArgumentException: Connection to 192.168.78.125:6379 not allowed. This connection point is not known in the cluster view
```

For details, see [Connection to X not allowed. This connection point is not known in the cluster view.](#)

Analysis

Specification modification process of a Redis Cluster instance:

After being started, the client obtains the cluster node topology by using the **Cluster Nodes** command based on RESP2, and maintains the topology in its in-memory data structure.

For data access, the client uses the CRC16 algorithm to calculate the hash slot of a key, and automatically routes requests based on the topology and slot information stored in the memory.

If the number of shards changes during scaling, the topology and slot mapping changes. In this case, the client needs to automatically update the topology. Otherwise, the request route may fail or the route location may be incorrect. As a result, an error is reported during client connection.

For example, when the number of shards in a Redis Cluster instance changes from three to six, the topology and slot mapping changes as shown in the following figures.

Figure 6-5 A Redis Cluster instance before scaling up

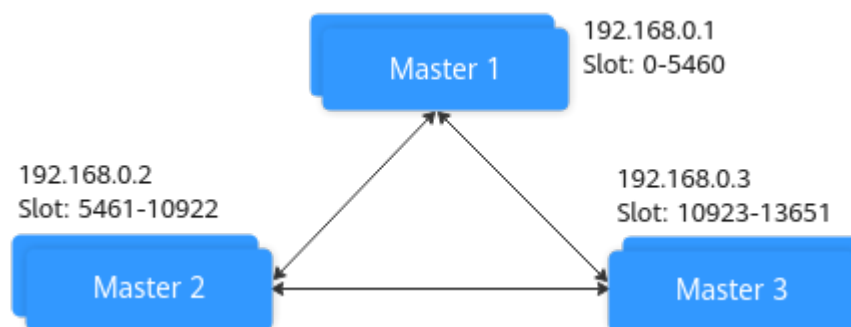
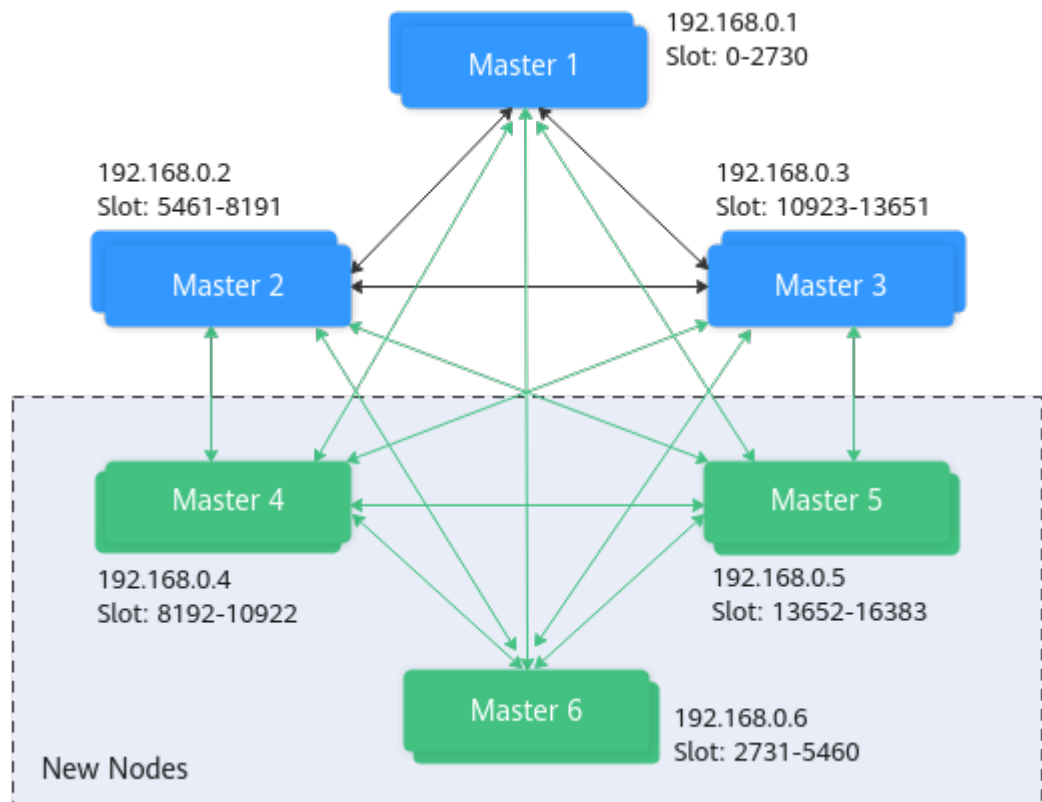


Figure 6-6 A Redis Cluster instance after scaling up



Solutions

Solution 1 (Recommended)

Enable automated topology refresh.

```
ClusterTopologyRefreshOptions topologyRefreshOptions = ClusterTopologyRefreshOptions.builder()
    // Periodic refresh: every time milliseconds.
    .enablePeriodicRefresh(Duration.ofMillis(time))
    // Triggers of adaptive refresh: MOVED redirection, ASK redirection, reconnection, unknown node
    // (since 5.1), and slot not in any of the current shards (since 5.2).
    .enableAllAdaptiveRefreshTriggers()
    .build();
```

For details, see [an example of using Lettuce to connect to a Redis Cluster instance](#).

NOTE

If you use Lettuce to connect to a Redis Cluster instance and automated refresh is not enabled, you need to restart the client after specification modification.

Solution 2

Disable validation of cluster node membership.

```
ClusterClientOptions clusterClientOptions = ClusterClientOptions.builder()
    .validateClusterNodeMembership(false)
    .build();
```

If `validateClusterNodeMembership` is `true`, check whether the current connection address is in the cluster topology obtained through **CLUSTER NODES**, before connecting to the cluster. If it is not in the topology, the error occurs.

 **NOTE**

Impact of disabling validation of cluster node membership:

- Lack of security breach detection.
- If automated topology refresh is disabled, a MOVED redirection request may be generated after the Redis Cluster specifications are changed and the shard quantity increases. Redirection increases the network load of the cluster and the time required to process a single request. If the shard quantity decreases, deleted shards cannot be connected.


6.11 Can I Expand a Single Shard of a Cluster Instance (Scale-Up)?

If you need to vertically expand a single shard of a cluster instance, contact technical support.

For more information about scale-up impacts, see [Impact of scaling](#).

Procedure

Step 1 Log in to the [DCS console](#).

Step 2 Click  in the upper left corner of the management console and select the region where your instance is located.

Step 3 In the navigation pane, choose **Cache Manager**.

Step 4 Choose **More > Modify Specifications** in the row containing the desired DCS instance.

Step 5 Set **Sharding** to **Customize** and select a **Shard Size**.

By default, if you are not on the whitelist for this feature, you can only select a value equal to or less than the current single shard size.

Figure 6-7 Selecting the size of a single shard

The screenshot displays the 'Current Specifications' section of a Redis Cluster instance. It includes details such as Region, Instance Name (dcs-...), Cache Engine (Basic | Redis 5.0), Billing Mode, Instance Type (Redis Cluster), Instance Name, CPU (x86), and Shard Size (1 GB). Below this, the 'Specifications' section shows Current Specification (4 GB), Replicas (2), and Shards (4).

Below the specifications, there are controls for 'Change By' (Specification, Replica quantity), 'Sharding' (Customize), and 'Shard Size' (1 GB). A table titled 'Instance Specification' is shown with a dropdown menu open, listing 1GB, 2GB, and 4GB options. The table columns are Cache Size, Shards, Max. Available Memory, CPU, and Max. Connections (Default/Limit).

Cache Size	Shards	Max. Available Memory	CPU	Max. Connections (Default/Limit)
8 GB	8	8 GB	x86	80,000/400,000
16 GB	16	16 GB	x86	160,000/800,000

----End

7 Data Backup, Export, and Migration

7.1 How Do I Export DCS Redis Instance Data?

- Master/standby, read/write splitting, and cluster instances:
These instances support backups. Perform the following operations to export data:
 - a. On the **Backups and Restorations** page, view the backup tasks.
 - b. If there is no backup, create a backup and download the backup file as prompted.

NOTE

If your DCS instances were created a long time ago, the versions of these instances may not be advanced enough to support some new functions (such as backup and restoration). You can contact technical support to upgrade your DCS instances. After the upgrade, you can back up and restore your instances.

- Single-node instances:
Single-node instances do not support the backup function. You can use `redis-cli` to export data to RDB files. This operation depends on **SYNC** command.
 - If the instance allows the **SYNC** command (such as a Redis 3.0 single-node instance), run the following command to export the instance data:
`redis-cli -h {source_redis_address} -p 6379 [-a password] --rdb {output.rdb}`
 - If the instance does not allow the **SYNC** command (such as a Redis 4.0 or 5.0 single-node instance), migrate the instance data to a master/standby instance and export the data by using the backup function.

7.2 Why Is Memory of a DCS Redis Instance Unchanged After Data Migration Using Rump, Even If No Error Message Is Returned?

For details on how to use Rump, see the [Data Migration Guide](#).

Possible causes:

- Rump does not support migration to cluster DCS instances.
- Commands are incorrectly run in Rump.

7.3 Can I Export Backup Data of DCS Redis Instances to RDB Files on the Console?

- DCS Redis 3.0 instances
No. On the console, backup data of a DCS Redis 3.0 instance can be exported only to AOF files. To export data to RDB files, run the following command in redis-cli:

```
redis-cli -h {redis_address} -p 6379 [-a password] --rdb {output.rdb}
```
- DCS Redis 4.0 and 5.0 instances
Yes. DCS Redis 4.0 and 5.0 instances support AOF and RDB persistence. You can back up data to RDB and AOF files on the console and download the files.

7.4 Why Are Processes Frequently Killed During Data Migration?

Possible cause: The memory is insufficient.

Solution: Expand the memory of the server on which the migration command is executed.

7.5 Where Are DCS Instance Backup Files Stored? How Are They Charged?

Backup files are stored to OBS. Currently, DCS and OBS do not charge for backups. In the future, a certain amount of fees may be charged based on the unified standard.

7.6 Is All Data in a DCS Redis Instance Migrated During Online Migration?

Migration among single-node, read/write splitting, or master/standby instances involves the full set of data. After the migration, a given key will remain in the same DB as it was before the migration. You can also specify a DB for migration.

By contrast, a cluster instance only has one DB, which is DB0 by default. During the migration, data in all slots of DB0 is migrated.

7.7 When Will AOF Rewrites Be Triggered?

AOF rewrites involve the following concepts:

- Rewrite window, which is currently 01:00 to 04:59
- Disk usage threshold, which is 50%
- Dataset memory, which is the percentage of memory that Redis dataset has used.

AOF rewrites are triggered in the following scenarios:

- If the disk usage reaches the threshold (regardless of whether the current time is within the rewrite window), rewrites will be triggered on instances whose AOF file size is larger than the dataset memory.
- If the disk usage is below the threshold and the current time is within the rewrite window, rewrites will be triggered on instances whose AOF file size is larger than the dataset memory multiplied by 1.5.
- If the disk usage is below the threshold but the current time is out of the rewrite window, rewrites will be triggered on instances whose AOF file size is larger than the instance's maximum memory multiplied by 4.5.

7.8 What Are the Common Causes of Redis Migration Failures?

- Check if a master/standby switchover occurred during the migration. If it occurred, contact technical support to temporarily disable master/standby switchover until the migration completes.
- For online migration, check whether the **SYNC** and **PSYNC** commands are disabled on the source Redis instance. If they are disabled, enable them to allow data synchronization.
- By default, a Proxy Cluster instance has only one database (DB0). Before you migrate data from a single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to [Enabling Multi-DB](#).
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to [Online Migration with Rump](#).

7.9 Can I Migrate Data to Multiple Target Instances in One Migration Task?

No. A migration task allows data to be migrated to only one target instance. To migrate data to multiple target instances, create multiple migration tasks.

7.10 How Do I Enable the SYNC and PSYNC Commands?

- Migration within DCS:
 - By default, the **SYNC** and **PSYNC** commands can be used when self-hosted Redis is migrated to DCS.
 - During online migration between DCS Redis instances in the same region under the same account, the **SYNC** and **PSYNC** commands are automatically enabled.
 - During online migration between DCS Redis instances in different regions or under different accounts within a region, the **SYNC** and **PSYNC** commands are not automatically enabled, and online migration cannot be used. You can migrate data using backup files.
- Migration from other cloud vendors to DCS:
 - Generally, cloud vendors disable the **SYNC** and **PSYNC** commands. If you want to use the online migration function on the DCS console, contact the O&M personnel of the source cloud vendor to enable the commands. For offline migration, you can import backup files.
 - If incremental migration is not required, you can perform full migration by referring to [Online Full Migration of Redis from Another Cloud with redis-shake](#). This method does not depend on **SYNC** and **PSYNC**.

7.11 Why Does Migration Task Creation Fail?

Possible causes:

1. The underlying resources are insufficient.
2. The specifications of the ECS used for the migration are insufficient.
3. The memory of the target Redis created before the migration is less than that of the source Redis.

In such cases, contact technical support.

7.12 Will the Same Keys Be Overwritten During Data Migration or Backup Import?

If the data exists on both the source and target instances, the target data is overwritten by the source data. If the data exists only on the target instance, the data will be retained.

Inconsistency between source and target data after the migration may be due to the target data that existed and was retained before migration.

8 Big/Hot Key Analysis and Expired Key Scan

8.1 What Are Big Keys and Hot Keys?

Term	Definition
Big key	<p>There are two types of big keys:</p> <ul style="list-style-type: none">• Keys that have a large value. If the size of a single String key exceeds 10 KB, or if the size of all elements of a key combined exceeds 50 MB, the key is defined as a big key.• Keys that have a large number of elements. If the number of elements in a key exceeds 5000, the key is defined as a big key.
Hot key	<p>A key is defined as a hot key if it is frequently requested or if it occupies a large number of resources. For example:</p> <ul style="list-style-type: none">• In a cluster instance, a shard processes 10,000 requests per second, among which 3000 are performed on the same key.• In a cluster instance, a shard uses a total of 100 Mbits/s inbound and outbound bandwidth, among which 80 Mbits/s is used by the HGETALL operation on a Hash key.

8.2 What Is the Impact of Big Keys or Hot Keys?

Category	Impact
Big key	<p>Instance specifications fail to be modified.</p> <p>Specification modification of a Redis Cluster instance involves rebalancing (data migration between nodes). Redis has a limit on key migration. If the instance has any single key bigger than 512 MB, the modification will fail when big key migration between nodes times out. The bigger the key, the more likely the migration will fail.</p>
	<p>Data migration fails.</p> <p>During data migration, if a key has many elements, other keys will be blocked and will be stored in the memory buffer of the migration ECS. If they are blocked for a long time, the migration will fail.</p>
	<p>Cluster shards are unbalanced.</p> <ul style="list-style-type: none"> • The memory usage of shards is unbalanced. For example, if a shard uses a large memory or even uses up the memory, keys on this shard are evicted, and resources of other shards are wasted. • The bandwidth usage of shards is unbalanced. For example, flow control is repeatedly triggered on a shard.
	<p>Latency of client command execution increases.</p> <p>Slow operations on a big key block other commands, resulting in a large number of slow queries.</p>
	<p>Flow control is triggered on the instance.</p> <p>Frequently reading data from big keys exhausts the outbound bandwidth of the instance, triggering flow control. As a result, a large number of commands time out or slow queries occur, affecting services.</p>
	<p>Master/standby switchover is triggered.</p> <p>If the high-risk DEL operation is performed on a big key, the master node may be blocked for a long time, causing a master/standby switchover.</p>
Hot key	<p>Cluster shards are unbalanced.</p> <p>If only the shard where the hot key is located is busy processing service queries, there may be performance bottlenecks on a single shard, and the compute resources of other shards may be wasted.</p>

Category	Impact
	<p>CPU usage surges.</p> <p>A large number of operations on hot keys may cause high CPU usage. If the operations are on a single cluster shard, the CPU usage of the shard where the hot key is located will surge. This will slow down other requests and the overall performance. If the service volume increases sharply, a master/standby switchover may be triggered.</p>
	<p>Cache breakdown may occur.</p> <p>If Redis cannot handle the pressure on hot keys, requests will hit the database. The database may break down as its load increases dramatically, affecting other services.</p>

8.3 How Do I Avoid Big Keys and Hot Keys?

- **Keep the size of Strings within 10 KB and the quantity of Hashes, Lists, Sets, or Zsets within 5000.**
- When naming keys, use the service name abbreviation as the prefix and do not use special characters such as spaces, line brakes, single or double quotation marks, and other escape characters.
- Do not rely too much on Redis transactions.
- The performance of short connections is poor. Use clients with connection pools.
- Do not enable data persistence if you use Redis just for caching and can tolerate data losses.
- For details about how to optimize big keys and hot keys, see the following table.

Category	Method
Big key	<p>Split big keys.</p> <p>Scenarios:</p> <ul style="list-style-type: none"> • If the big key is a String, you can split it into several key-value pairs and use MGET or a pipeline consisting of multiple GET operations to obtain the values. In this way, the pressure of a single operation can be split. For a cluster instance, the operation pressure can be evenly distributed to multiple shards, reducing the impact on a single shard. • If the big key contains multiple elements, and the elements must be operated together, the big key cannot be split. You can remove the big key from Redis and store it on other storage media instead. This scenario should be avoided by design. • If the big key contains multiple elements, and only some elements need to be operated each time, separate the elements. Take a Hash key as an example. Each time you run the HGET or HSET command, the result of the hash value modulo N (customized on the client) determines which key the field falls on. This algorithm is similar to that used for calculating slots in Redis Cluster. <p>Store big keys on other storage media.</p> <p>If a big key cannot be split, it is not suitable to be stored in Redis. You can store it on other storage media, such as SFS or other NoSQL databases, and delete the big key from Redis.</p> <p>CAUTION</p> <p>Do not use the DEL command to delete big keys. Otherwise, Redis may be blocked or even a master/standby switchover may occur.</p> <p>Set appropriate expiration and delete expired data regularly.</p> <p>Appropriate expiration prevents expired data from remaining in Redis. Due to Redis's lazy free, expired data may not be deleted in time. If this occurs, scan expired keys.</p>
Hot key	<p>Split read and write requests.</p> <p>If a hot key is frequently read, configure read/write splitting on the client to reduce the impact on the master node. You can also add replicas to meet the read requirements, but there cannot be too many replicas. In DCS, replicas replicate data from the master in parallel. The replicas are independent of each other and the replication delay is short. However, if there is a large number of replicas, CPU usage and network load on the master node will be high.</p>

Category	Method
	<p>Use the client cache or local cache.</p> <p>If you know what keys are frequently used, you can design a two-level cache architecture (client/local cache and remote Redis). Frequently used data is obtained from the local cache first. The local cache and remote cache are updated with data writes at the same time. In this way, the read pressure on frequently accessed data can be separated. This method is costly because it requires changes to the client architecture and code.</p>
	<p>Design a circuit breaker or degradation mechanism.</p> <p>Hot keys can easily result in cache breakdown. During peak hours, requests are passed through to the backend database, causing service avalanche. To ensure availability, the system must have a circuit breaker or degradation mechanism to limit the traffic and degrade services if breakdown occurs.</p>

8.4 How Do I Analyze the Hot Keys of a DCS Redis 3.0 Instance?

DCS for Redis 3.0 does not support hot key analysis on the console. Alternatively, you can use the following methods to analyze hot keys:

- Method 1: Analyze the service structure and service implementation to discover possible hot keys.

For example, hot keys can easily be found in the service code during flash sales or user logins.

Advantage: Simple and easy to implement.

Disadvantage: Requires familiarity with the service code. In addition, the analysis become more difficult as the service scenarios become more complex.
- Method 2: Collect key access statistics in the client code to discover hot keys.

Disadvantage: Requires intrusive code modification.
- Method 3: Capture and analyze packets.

Advantage: Simple and easy to implement.

8.5 How Do I Detect Big Keys and Hot Keys in Advance?

Method	Description
Through Big Key Analysis and Hot Key Analysis on the DCS console	See Analyzing Big Keys and Hot Keys .

Method	Description
By using the bigkeys and hotkeys options on redis-cli	<ul style="list-style-type: none"> redis-cli uses the bigkeys option to traverse all keys in a Redis instance and returns the overall key statistics and the biggest key of six data types: Strings, Lists, Hashes, Sets, Zsets, and Streams. The command is redis-cli -h <Instance connection address> -p <Port number> -a <Password> --bigkeys. In Redis 4.0 and later, you can use the hotkeys option to quickly find hot keys in redis-cli. Run this command during service running to find hot keys: redis-cli -h <Instance connection address> -p <Port number> -a <Password> --hotkeys. The hot key details can be obtained from the summary part in the returned result.
Searching for big keys using Redis commands	<p>If there is a pattern of big keys, for example, the prefix is cloud:msg:test, you can use a program to scan for keys that match the prefix, and then run commands to query the number of members in the key and query the key sizes to find big keys.</p> <ul style="list-style-type: none"> Commands for querying the number of members: LLEN, HLEN, XLEN, ZCARD, SCARD Commands for querying the memory usage of keys: DEBUG OBJECT, MEMORY USAGE <p>CAUTION This method consumes a large number of computing resources. To ensure service running, do not use this method on instances with heavy service pressure.</p>
Searching for big keys using redis-rdb-tools	<p>redis-rdb-tools is an open-source tool for analyzing Redis RDB snapshot files. You can use it to analyze the memory usage of all keys in a Redis instance.</p> <p>To use this method, you must export the RDB file of an instance on the Backups & Restorations page of the DCS console.</p> <p>CAUTION This method does not affect service running, but is not as timely as online analysis.</p>

Hot key analysis is not supported by DCS Redis 3.0 instances. However, you can **configure alarms** to detect hot keys.

- Configure alarm rules for the **Memory Usage** metric of the instance nodes. If a node has a big key, the memory usage of the node is much higher than that of other nodes. In this case, an alarm is triggered to help you find the potentially problematic key.
- Configure alarm rules for the **Maximum Inbound Bandwidth, Maximum Outbound Bandwidth**, and **CPU Usage** metrics of the instance nodes.

If a node has a hot key, the bandwidth and CPU usage of the node is much higher than that of other nodes. In this case, an alarm is triggered to help you find the potentially problematic key.

8.6 How Does DCS Delete Expired Keys?

Question

What are the rules for scheduled deletion of expired keys on a daily basis? Can I customize the rules?

Mechanisms for Deleting Expired Keys

- **Lazy free deletion:** The deletion strategy is controlled in the main I/O event loop. Before a read/write command is executed, a function is called to check whether the key to be accessed has expired. If it has expired, it will be deleted and a response will be returned indicating that the key does not exist. If the key has not expired, the command execution resumes.
- **Scheduled deletion:** A time event function is executed at certain intervals. Each time the function is executed, a random collection of keys are checked, and expired keys are deleted.

NOTE

To avoid prolonged blocks on the Redis main thread, not all keys are checked in each time event. Instead, a random collection of keys are checked each time. As a result, the memory used by expired keys cannot be released quickly.

Solutions

- Configure scheduled hot key analysis tasks by referring to [Hot Key Analysis](#), or use the **SCAN** command to traverse all keys on a scheduled basis and remove expired keys from the memory.
- Configure a scheduled task to scan all master nodes of the instance. All keys will be scanned, and Redis will determine whether the keys have expired. Expired keys will be released. For details, see [Scanning Expired Keys](#).

How Do I Know Which Expired Keys Have Been Deleted?

Deleted expired keys cannot be queried.

8.7 How Long Are Keys Stored? How Do I Set Key Expiration?

- **Key storage duration**
 - Keys that do not have an expiration are stored permanently.
 - Keys that have an expiration are deleted after they expire. For details, see [Scanning Expired Keys](#).
 - To remove the expiration set for a key, run the **PERSIST** command.

- Setting key expiration
You can run the **EXPIRE** or **PEXPIRE** command to set the key expiration time. For example, if you run **expire key1 100**, key1 will expire in 100 seconds. If you run **pexpire key2 1800**, key2 will expire in 1800 milliseconds.
EXPIRE sets key expiration in seconds, and **PEXPIRE** sets key expiration in milliseconds.

8.8 Why Does Memory Usage Decrease After Big Key Analysis Is Performed on Redis?

Big key analysis only queries keys that occupy a large space and does not delete the keys. Due to the lazy free deletion mechanism, keys are not deleted immediately upon expiration, unless they are accessed or identified. During a big key analysis, all keys are traversed, and the expired keys will be identified and then deleted. As a result, memory usage decreases after a big key analysis is completed.

For details about how expired keys are deleted and how to scan for expired keys, see [Scanning Expired Keys](#).

9 Redis Commands

9.1 Does DCS for Redis Support Command Audits?

No. To ensure high-performance read and write operations, DCS for Redis does not audit commands. Commands are not printed.

9.2 How Do I Clear Redis Data?

Exercise caution when clearing data.

- Redis 3.0
Data of a DCS Redis 3.0 instance cannot be cleared on the console, and can only be cleared by the **FLUSHDB** or **FLUSHALL** command in redis-cli.
Run the **FLUSHALL** command to clear all the data in the instance.
Run the **FLUSHDB** command to clear the data in the currently selected database.
- Redis 4.0 or later
To clear data of a DCS Redis 4.0 or later instance, you can run the **FLUSHDB** or **FLUSHALL** command in redis-cli, use the data clearing function on the DCS console, or run the **FLUSHDB** command on Web CLI.
To clear data of a Redis Cluster instance, run the **FLUSHDB** or **FLUSHALL** command on every shard of the instance. Otherwise, data may not be completely cleared.

NOTE

- Currently, the data clearing function and Web CLI access on the console are supported only by DCS Redis 4.0 or later instances.
- When you run the **FLUSHDB** command on Web CLI, only one shard is cleared at a time. If there are multiple shards, connect to the master node of each shard and run the **FLUSHDB** command separately.
- Redis Cluster data cannot be cleared by using Web CLI.

9.3 How Do I Find Specified Keys and Traverse All Keys?

Finding Specified Keys

Big key and hot key analysis does not support key searching with specified conditions. To find keys with the specified prefix or suffix, use the **SCAN** command.

For example, to search for keys that contain the letter *a* in a Redis instance, run the following command in redis-cli:

```
./redis-cli -h {redis_address} -p {port} [-a password] --scan --pattern '*a*'
```

Traversing All Keys

Do not use the **KEYS** command to traverse all keys of an instance because the **KEYS** command is complex and may block Redis. To traverse all keys in a DCS Redis instance, run the following command in redis-cli:

```
./redis-cli -h {redis_address} -p {port} [-a password] --scan --pattern '*'
```

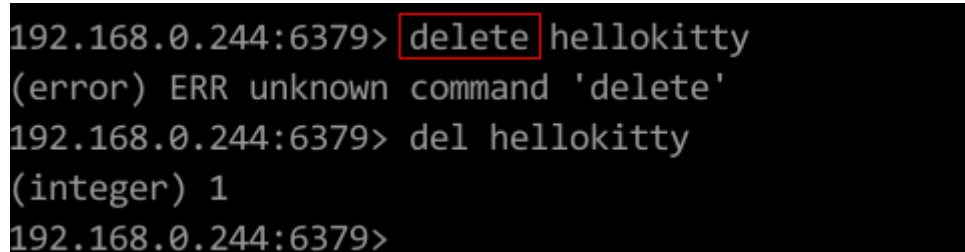
For details about the **SCAN** command, visit the [Redis official website](#).

9.4 Why Do I Fail to Execute Some Redis Commands?

Possible causes include the following:

- The command is spelled incorrectly.

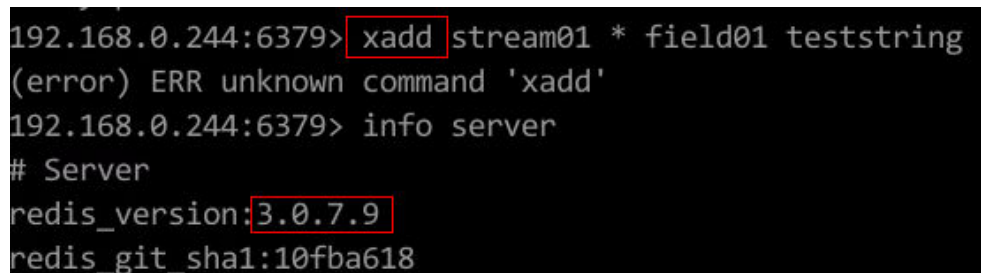
As shown in the following figure, the error message is returned because the correct command for deleting a key should be **del**.



```
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>
```

- A command available in a higher Redis version is run in a lower Redis version.

As shown in the following figure, the error message is returned because a stream command (available in Redis 5.0) is run in Redis 3.0.



```
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
```

- The command is disabled in DCS.
For security purposes, some Redis commands are disabled in DCS. For details about disabled and restricted Redis commands, see [Command Compatibility](#).
- The command cannot be executed in Web CLI.
In addition to the disabled and restricted Redis commands, the **KEYS** commands is also restricted in Web CLI.
- The LUA script fails to be executed.
For example, the error message "ERR unknown command 'EVAL'" indicates that your DCS Redis instance is of a lower version that does not support the LUA script. In this case, submit a service ticket for the instance to be upgraded.
- The **CLIENT SETNAME** and **CLIENT GETNAME** commands fail to be executed.
The DCS Redis instance is of a lower version that does not support these commands. In this case, submit a service ticket for the instance to be upgraded.
- The following commands are disabled for **cluster** DCS Redis instances created before July 10, 2018. You can upgrade such an instance by submitting a service ticket.
SINTER, SDIFF, SUNION, PFCOUNT, PFMERGE, SINTERSTORE, SUNIONSTORE, SDIFFSTORE, SMOVE, BLPOP, BRPOP, BRPOPLPUSH, ZUNIONSTORE, ZINTERSTORE, EVAL, EVALSHA, BITOP, RENAME, RENAMENX, RPOPLPUSH, MSETNX, SCRIPT LOAD, SCRIPT KILL, SCRIPT EXISTS, SCRIPT FLUSH

9.5 Why is "permission denied" Returned When I Run the Keys Command in Web CLI?

The **KEYS** command is disabled in Web CLI. This command can only be run in redis-cli.

9.6 How Do I Rename High-Risk Commands?

Currently, you can only rename the **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, **HGETALL**, **SCAN**, **HSCAN**, **SSCAN**, and **ZSCAN** commands. For details, see [Renaming Commands](#).

NOTE

- Currently, Redis does not support disabling of commands. To avoid risks when using the preceding commands, rename them. For details about the supported and disabled commands in DCS, see [Command Compatibility](#).
- After you confirm the modification, the system will restart the instance. The new commands take effect after the restart. Remember the new command names because they will not be displayed on the console for security purposes.
- Command renaming is available only to Redis 4.0 and later and not to Redis 3.0.

9.7 Does DCS for Redis Support Pipelining?

Yes.

For Redis Cluster instances, ensure that all commands in a pipeline are executed on the same shard.

9.8 Does DCS for Redis Support the INCR and EXPIRE Commands?

Yes.

For more information about Redis command compatibility, see [Redis Command Compatibility](#).

9.9 Why Does a Redis Command Fail to Take Effect?

Run the command in redis-cli to check whether the command takes effect.

The following describes two scenarios:

- Scenario 1: Set and query the value of a key to check whether the **SET** and **GET** commands work.

The **SET** command is used to set the string value. If the value is not changed, run the following commands in redis-cli to access the instance:

```
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>
```

- Scenario 2: If the timeout set using the **EXPIRE** command is incorrect, perform the following operations:

Set the timeout to 10 seconds and run the **TTL** command to view the remaining time. As shown in the following example, the remaining time is 7 seconds.

```
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>
```

NOTE

Redis clients (including redis-cli, Jedis clients, and Python clients) communicate with Redis server using a binary protocol.

If Redis commands are run properly in redis-cli, the problem may lie in the service code. In this case, create logs in the code for further analysis.

9.10 Is There a Time Limit on Executing Redis Commands? What Will Happen If a Command Times Out?

Redis command timeouts can be controlled on the client end or server end.

- Timeouts on the client end are controlled in the client code. You can determine the timeouts that suit service needs. For example, if you use Lettuce, a Java client, configure the **timeout** parameter.
- On the server end, the **timeout** parameter is set to **0** by default, indicating that connections will never be terminated. Modify the parameter setting by referring to [Modifying Configuration Parameters](#).

9.11 Can I Configure Redis Keys to Be Case-Insensitive?

No. Like in open-source Redis, keys in DCS for Redis are case-sensitive and cannot be configured to be case-insensitive.

9.12 Can I View the Most Frequently Used Redis Commands?

No. Redis does not record commands and does not support viewing the most frequently used commands.

9.13 Common Web CLI Errors

1. ERR Wrong number of arguments for 'xxx' command
This error indicates that the executed Redis command has a parameter error (syntax error). Rewrite the command by referring to the open-source Redis command protocol.
2. ERR unknown command 'xxx'
This error indicates that the command is unknown or is not a valid command defined by Redis. Rewrite the command by referring to the open-source Redis command protocol.
3. ERR Unsupported command: 'xxx'
This error indicates that the command is disabled for DCS Redis instances. For details, see [Web CLI Commands](#).

10 Monitoring and Alarm

10.1 Why Is CPU Usage of a DCS Redis Instance 100%?

Symptom

The CPU usage of a Redis instance increases dramatically within a short period of time. If the CPU usage is too high, connections may time out, and master/standby switchover may be triggered.

Possible Causes

1. The service QPS is high. In this case, refer to [Checking QPS](#).
2. Resource-consuming commands, such as **KEYS**, were used. In this case, refer to [Locating and Disabling CPU-Intensive Commands](#).
3. Redis rewrite was triggered. In this case, refer to [Checking Redis Rewrite](#).

Checking QPS

On the **Cache Manager** page of the DCS console, click an instance to go to the instance details page. On the left menu, choose **Performance Monitoring** and then view the **Ops per Second** metric.

If the QPS is high, optimize customer services or [modifying instance specifications](#). For details about the QPS supported by different instance specifications, see [DCS Instance Specifications](#).

Locating and Disabling CPU-Intensive Commands

Resource-consuming commands (commands with time complexity $O(N)$ or higher), such as **KEYS**, are used. Generally, the higher the time complexity, the more resources a command uses. As a result, the CPU usage is high, and a master/standby switchover can be easily triggered. For details about the time complexity of each command, visit the [Redis official website](#). In this case, use the **SCAN** command instead or disable the **KEYS** command.

- Step 1** On the **Performance Monitoring** page of the DCS console, locate the period when the CPU usage is high.

Step 2 Use the following methods to find the commands that consume a large number of resources.

- Redis logs queries that exceed a specified execution duration. You can find the commands that consume a large number of resources by analyzing the slow queries and their execution duration. For details, see [Viewing Redis Slow Queries](#).
- Use the instance diagnosis function to analyze the execution duration percentage of different commands during the period when the CPU usage is high. For details, see [Diagnosing an Instance](#).

The screenshot displays the 'Instance Diagnosis' interface for a Redis instance. It includes a sidebar with navigation options like 'Basic Information', 'Performance Monitoring', and 'Instance Configuration'. The main area shows 'Tested Objects (2)' with a search bar and a tree view of nodes. Below this is a 'Test History' table. The 'Command Execution Latency' section is highlighted, showing a table of command performance metrics.

Command	Average Latency (ms)	Duration Percentage	Execution TL	Total Latency (ms)
info	0.06	77.67%	68	4.069
config	0.017	0.32%	1	0.017
publish	0.004	13.42%	171	0.703
subscribe	0.003	0.17%	3	0.009
slowlog	0.002	0.08%	2	0.004

Step 3 Resolve the problem.

- Evaluate and disable high-risk and high-consumption commands, such as **FLUSHALL**, **KEYS**, and **HGETALL**.
- Optimize services. For example, avoid frequent data sorting operations.
- (Optional) Perform the following operations to adjust instances based on service requirements:
 - Change the instance type to read/write splitting to separate read and write requests from high-consumption commands or applications.
 - Scale up the instance.

----End

Checking Redis Rewrite

AOF persistence, which is enabled by default for master/standby and cluster DCS Redis instances, takes place in the following scenarios:

- If a small amount of data is written and the AOF file is not large, AOF rewrite is performed from 01:00 to 04:00 in the morning every day, and CPU usage may suddenly spike during this period.
- When a large amount of data is written and the AOF file size exceeds the threshold (three to five times the DCS instance capacity), AOF rewrite is automatically triggered in the background regardless of the current time.


Redis rewrite is performed by running the **BGSAVE** or **BGREWRITEAOF** command, which may consume many CPU resources (see [the discussion](#)). **BGSAVE** and **BGREWRITEAOF** commands need to fork(), resulting in CPU usage spikes within a short period of time.

If persistence is not required, disable it by changing the value of **appendonly** to **no** on the **Parameters** page of the instance. However, if you disable persistence, data loss may occur due to a lack of data flushing to disk in extreme situations.

10.2 How Do I View Current Concurrent Connections and Maximum Connections of a DCS Redis Instance?

Viewing Concurrent Connections of a DCS Redis Instance

The number of real-time connections received by a DCS instance is a metric that can be viewed on the console. For details on how to view the metrics, see [Viewing DCS Monitoring Metrics](#).

On the Cloud Eye console, find the **Connected Clients** metric. Click  to view monitoring details on an enlarged graph.

Specify a time range to view the metric in a specific monitoring period. For example, you can select a 10-minute period to view the number of connections received during the period. On the graph, you can view the trend and the total number of connections received during the period.

NOTE

The **Connected Clients** metric means the number of connected clients. It includes connections established for system monitoring, configuration synchronization, and services, but excludes connections from replicas.

You can also view client session information by using the [session management](#) function on the console.

Viewing or Modifying the Maximum Connections of an Instance

You can view the default and maximum allowed number of connections on the instance creation page or in the [document](#).

After an instance is created, you can view or change the value of **maxclients** (the maximum number of connections) on the **Instance Configuration > Parameters** page of the instance. (Proxy Cluster instances do not have this parameter.)

If the limit is exceeded, excess requests will be rejected and connections will time out.

10.3 What Should I Do If the Monitoring Data of a DCS Redis Instance Is Abnormal?

If you have any doubt on the monitoring data of a DCS Redis instance, you can access the instance through redis-cli and run the **INFO ALL** command to view the metrics. For details about the output of the **INFO ALL** command, see <https://redis.io/docs/latest/commands/info/>.

10.4 Why Is Used Memory Greater Than Available Memory?

For single-node and master/standby DCS instances, the used instance memory is measured by the Redis-server process. For cluster DCS instances, the used cluster memory is the sum of used memory of all shards in the cluster.

Due to the internal implementation of the open-source redis-server, the used instance memory is normally slightly higher than the available instance memory.

Redis allocates memory using zmalloc. It does not check whether used_memory exceeds max_memory every time the memory is allocated. Instead, it checks whether the current used_memory exceeds max_memory at the beginning of a periodic task or command processing. If used_memory exceeds max_memory, eviction is triggered. Therefore, the restrictions of the max_memory policy are not implemented in real time or rigidly. A case in which the used_memory is greater than the max_memory may occur occasionally.

10.5 Why Does Bandwidth Usage Exceed 100%?

The basic information about the bandwidth usage metric is as follows.

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Raw Data)
bandwidth_usage	Bandwidth Usage	Percentage of the used bandwidth to the maximum bandwidth limit	0-200%	Monitored object: Redis 4.0 and later Redis Server of a master/standby, read/write splitting, or cluster instance Dimension: dcs_cluster_node	1 minute

$\text{Bandwidth usage} = (\text{Input flow} + \text{Output flow}) / (2 \times \text{Maximum bandwidth}) \times 100\%$

According to the formula, the bandwidth usage counts in the input flow and output flow, which include the traffic for replication between the master and replicas. Therefore, the total bandwidth usage is larger than the normal service traffic, and may exceed 100%.

If the value of the **Flow Control Times** metric is larger than 0, the maximum bandwidth has been reached and flow control has been performed.

However, flow control decisions are made without considering the traffic for replication between the master and replicas. Therefore, sometimes the bandwidth usage exceeds 100% but the number of flow control times is 0.

10.6 Why Is the Rejected Connections Metric Displayed?

If the **Rejected Connections** metric is displayed, check if the number of connected clients exceeds the maximum allowed number of connections of the instances.

- To check the maximum allowed number of connections, go to the **Parameters** tab page of the instance and check the value of the **maxclients** parameter. (Proxy Cluster instances do not have this parameter. You can view the maximum number of connections on the instance creation page.)
- To check the current number of connections, go to the **Performance Monitoring** tab page of the instance and check the **Connected Clients** metric.

If the current number of connections reaches the upper limit, you can adjust the value of **maxclients**. If the value of **maxclients** can no longer be increased, increase the instance specifications.

10.7 Why Is Flow Control Triggered? How Do I Handle It?

Flow control is triggered when the traffic used by a Redis instance in a period exceeds the maximum bandwidth. Connections may be discarded due to flow control, resulting in high service latency and client connection exceptions.

NOTE

For details about the maximum allowed bandwidth, see the "Assured/Maximum Bandwidth" column of different instance types listed in [DCS Instance Specifications](#).

Even if the bandwidth usage is low, flow control may still be triggered. The real-time bandwidth usage is reported once in each reporting period. Flow controls are checked every second. The traffic may surge within seconds and then fall back between reporting periods. By the time the bandwidth usage is reported, it may have already restored to the normal level.

For master/standby instances:

- If flow control is always triggered when the bandwidth usage is low, there may be service microbursts or big or hot keys. In this case, check for big or hot keys.
- If the bandwidth usage remains high, the bandwidth limit may be exceeded. In this case, expand the capacity. Larger capacity supports higher bandwidth.

For cluster instances:

- If flow control is triggered only on one or a few shards, the shards may have big or hot keys.
- If flow control or high bandwidth usage occurs on all or most shards at the same time, bandwidth usage of the instance has reached the limit. In this case, expand the instance capacity.

 **NOTE**

- Perform big key and hot key analysis on the DCS console, and take measures accordingly. For details, see [Analyzing Big Keys and Hot Keys](#).
- Running commands (such as **KEYS**) that consume lots of resources may cause high CPU and bandwidth usage. As a result, flow control is triggered.
- You can perform [Temporary bandwidth expansion](#) on DCS Redis instances on the console to temporarily solve the problem that the bandwidth exceeds the threshold during peak hours.

11 Master/Standby Switchover

11.1 When Does a Master/Standby Switchover Occur?

A master/standby switchover may occur in the following scenarios:

- A master/standby switchover operation is initiated on the DCS Console.
- A master/standby switchover will be triggered when the master node of a master/standby instance fails.

For example, if commands (such as **KEYS**) that consume a lot of resources are used or logs are aged and deleted in batches, the CPU usage will surge, triggering a master/standby switchover.

- If you restart a master/standby instance on the DCS console, a master/standby switchover will be triggered.
- If you scale up a single-node, read/write splitting, or master/standby instance, a master/standby switchover will be triggered.

During scale-up, a new standby node with the new specifications is created. After full and incremental data on the master node is synchronized to the standby node, a master/standby switchover is performed and the original node is deleted.

After a master/standby switchover occurs, you will receive a notification. Check whether the client services are running properly. If not, check whether the TCP connection is normal and whether it can be re-established after the master/standby switchover to restore the services.

11.2 How Does Master/Standby Switchover Affect Services?

If a fault occurs in a master/standby or cluster DCS instance, a failover is triggered automatically. Services may be interrupted for less than half a minute during exception detection and failover.

11.3 Does the Client Need to Switch the Connection Address After a Master/Standby Switchover?

No. If the master node fails or a master/standby switchover is performed, the standby node will be promoted to master and takes the original IP address.

11.4 How Does Redis Master/Standby Replication Work?

Redis master/standby instances are also called master/slave instances. Generally, updates to the master cache node are automatically and asynchronously replicated to the standby cache node. This means that data in the standby cache node may not always be consistent with data in the master cache node. The inconsistency is typically seen when the I/O write speed of the master node is faster than the synchronization speed of the standby node or a network latency occurs between the master and standby nodes. If a failover happens when some data is not yet replicated to the standby node, such data may be lost after the failover.

12 Instance Creation and Permissions

12.1 Why Do I Fail to Create a DCS Redis or Memcached Instance?

- The subnet does not have sufficient IP addresses.
Analysis: Each node in a DCS instance must be assigned an IP address. Therefore, a single-node instance requires one IP address, a master/standby instance requires two IP addresses, and a cluster instance requires multiple IP addresses.
Solution: Create the instance in a different subnet within the VPC or release IP addresses in the current subnet.
- The IAM user does not have the permissions required to create an instance.
Analysis: The group to which the user belongs must be granted the **DCS FullAccess** policy or **DCS Administrator** role or other policies containing the permissions required for creating DCS instances.
Solution: Create a DCS instance as the administrator.

12.2 Why Can't I View the Subnet and Security Group Information When Creating a DCS Instance?

This may be because you do not have the **Server Administrator** and **VPC Administrator** roles. For details on how to add user permissions, see [Modifying User Group Permissions](#).

12.3 Why Can't I Select the Required Enterprise Project When Creating a DCS Instance?

Symptom

The desired enterprise project is not displayed during instance creation.

Cause

The user group does not have DCS permissions in the desired enterprise project.

Solution

1. Log in to the DCS console.
2. In the upper right corner, choose **Enterprise > Project Management**. On the displayed page, click **View Resource** in row containing the desired enterprise project.
3. Click the **Permissions** tab. Then, click **Authorize User Group**.

NOTE

Click **Authorize User Group** to grant permissions to a user group, or click **Authorize User** to grant permissions to a user.

4. Click **Authorize** in the row containing the user or user group to which you want to grant permissions.
5. Search for and select the **DCS FullAccess** policy, click **Next**, and click **OK**.

For more information about DCS permissions policies, see [Permissions Management](#).

NOTE

If you configure both the **DCS UserAccess** (containing deny statements) and **DCS FullAccess** policies, you cannot create, modify, delete, or scale DCS instances because deny statements will take precedence. To perform the operations allowed by **DCS FullAccess**, delete **DCS UserAccess** first.

12.4 Why Can't an IAM User See a New DCS Redis Instance?

Symptom

An IAM user cannot see a newly created DCS Redis instance.

Possible Cause

The IAM user does not have permissions for the enterprise project to which the new instance belongs.

Solution

1. Log in to the DCS console.
2. In the upper right corner, choose **Enterprise > Project Management**. On the displayed page, click **View Resource** in row containing the desired enterprise project.
3. Click the **Permissions** tab. Then, click **Assign Permissions** on the **User Groups** tab.
4. Select user groups you want to assign permissions to, and click **Next**.

5. Select **DCS UserAccess** and click **OK**.

13 Memcached Usage

13.1 Can I Dump DCS Memcached Instance Data for Analysis?

No.

13.2 What Memcached Version Is Compatible with DCS for Memcached?

DCS for Memcached is based on Redis 3.0 and is compatible with Memcached 1.5.1.

13.3 What Data Structures Does DCS for Memcached Support?

Only the key-value structure is supported.

13.4 Does DCS for Memcached Support Public Access?

No.

If public access is disabled for a DCS instance, you cannot access it in local environments and can only access it through an ECS that is in the same VPC as the instance. VPCs are used to ensure network security of public cloud services.

During application development and debugging, you can connect to a DCS instance from your local environment by using an ECS that can communicate with your instance for forwarding your requests. For details, see [Using SSH Tunneling for Public Access to a DCS Instance](#).

13.5 Can I Modify Configuration Parameters of DCS Memcached Instances?

Parameter configuration is allowed only when DCS instances are in the **Running** state.

For details, see [Modifying Configuration Parameters](#).

13.6 What Are the Differences Between DCS for Memcached and Self-Hosted Memcached?

Table 13-1 describes the differences between DCS for Memcached and self-hosted Memcached.

Table 13-1 Comparing DCS for Memcached and self-hosted Memcached

Item	DCS for Memcached	Self-Hosted Memcached
Deployment	Easy to deploy. DCS for Memcached can be used right out of the box without requiring you to worry about the hardware or software.	Involves complicated operations and settings.
Availability	Master/Standby instances use hot standby to ensure stable services. If the master node is faulty, the standby cache node will automatically become the master node to prevent a single point of failure.	Requires additional configurations.
Security	Uses the VPC and security groups for network access security control.	Requires you to design and implement a security mechanism by yourself.
Scale-up	Supports online scale-up on the console.	Requires additional hardware and restarting your service.

13.7 What Policies Does DCS for Memcached Use to Deal with Expired Data?

DCS for Memcached allows you to set the expiration time for stored data based on service requirements. For example, you can set the **expire** time when performing the **add** operation.

```
>> help add
Synopsis: >> add <key> <value> <expire>

Options:
  • <key> (string, required)
    add key
  • <value> (string, required)
    add value
  • <expire> (string, required)
```

By default, data is not evicted from DCS Memcached instances. In the current version of DCS for Memcached, you can select an eviction policy.

For details about the six types of data eviction policies, see [What Are Redis Data Eviction Policies?](#)

13.8 How Should I Select AZs When Creating a DCS Memcached Instance?

Different AZs within a region do not differ in functions.

Generally, instance deployment within an AZ features lower network latency while cross-AZ deployment ensures disaster recovery. If your application requires lower network latency, choose single-AZ deployment.

DCS for Memcached supports cross-AZ deployment. When creating a DCS Memcached instance on the DCS console, you can select any AZ in the same region as your ECS for communication between your ECS and instance. For lower network latency, select the same AZ as your ECS.

Assume that you have an ECS that is in **AZ B** in the **CN-Hong Kong** region. When creating a DCS Memcached instance, you can select any AZ in **CN-Hong Kong**. If you select **AZ B** in **CN-Hong Kong**, your instance can communicate with your ECS with lower network latency.

Note that there may be only one available AZ due to insufficient resources when you create a DCS Memcached instance. This does not affect the normal use of DCS.