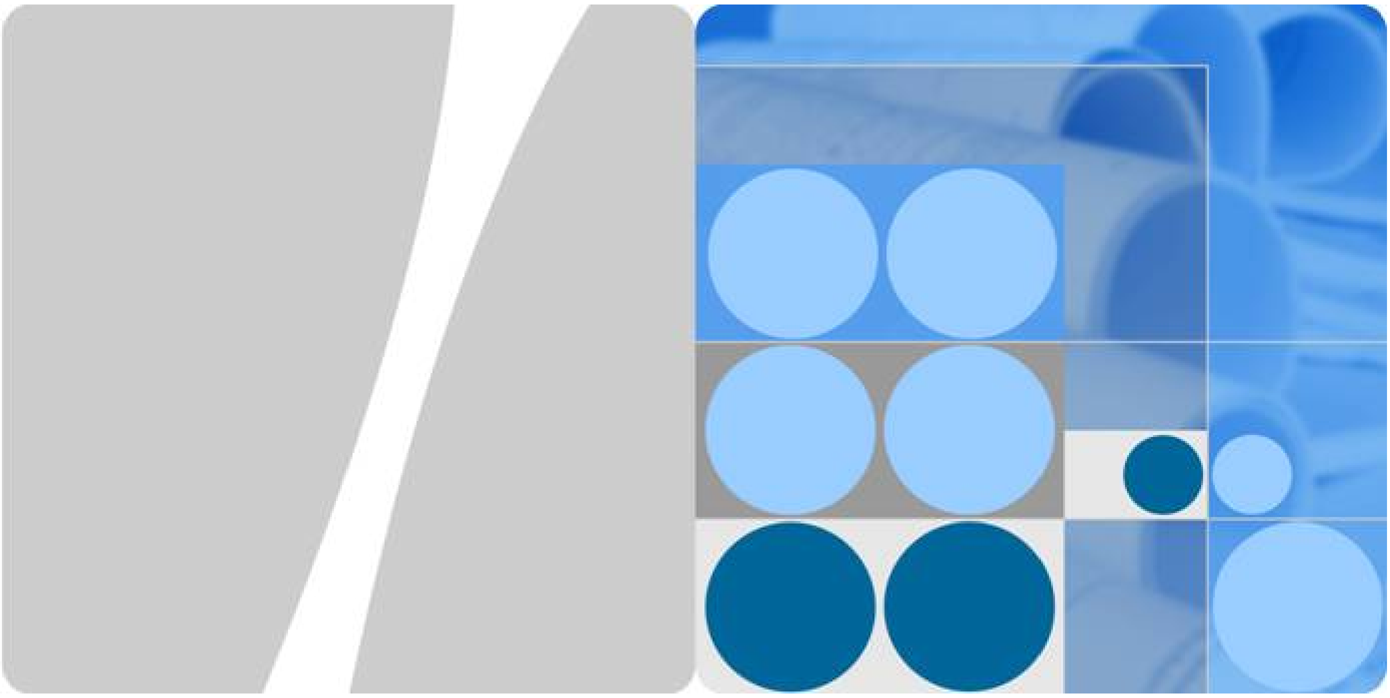


Distributed Cache Service

FAQs

Date **2020-03-17**

Contents

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	Supports up to 256 Redis databases.	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 Redis Versions

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

- **Version**

  DCS supports Redis 3.0, 4.0, and 5.0. **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 and Redis 5.0
Instance deployment mode	Based on VMs	Containerized based on physical servers
CPU architecture	x86	x86 and Arm
Time required for creating an instance	3–15 minutes Cluster instance: 10–30 minutes	8 seconds
QPS	100,000 QPS per node	100,000 QPS per node
Public network access	Supported	Not supported
Domain name connection	Supported in VPC	Supported in VPC
Visualized data management	Not supported	Provides Web CLI for Redis access and data management.
Instance type	Single-node, master/standby, and Proxy Cluster	Single-node, master/standby, and Redis Cluster

Feature	Redis 3.0	Redis 4.0 and Redis 5.0
Instance total memory	Ranges from 2 GB to 1 TB.	Regular specifications range from 2 GB to 1 TB. Small specifications of 128 MB, 256 MB, 512 MB, and 1 GB are also available for single-node and master/standby instances.
Capacity expansion/reduction	Online capacity expansion and reduction	Online capacity expansion and reduction
Backup and restoration	Supported for master/standby and cluster instances	Supported for master/standby and cluster 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, create 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 FLUSHAL command to clear its own data.
```

NOTE

To enable these configurations, submit a service ticket to contact technical support.

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

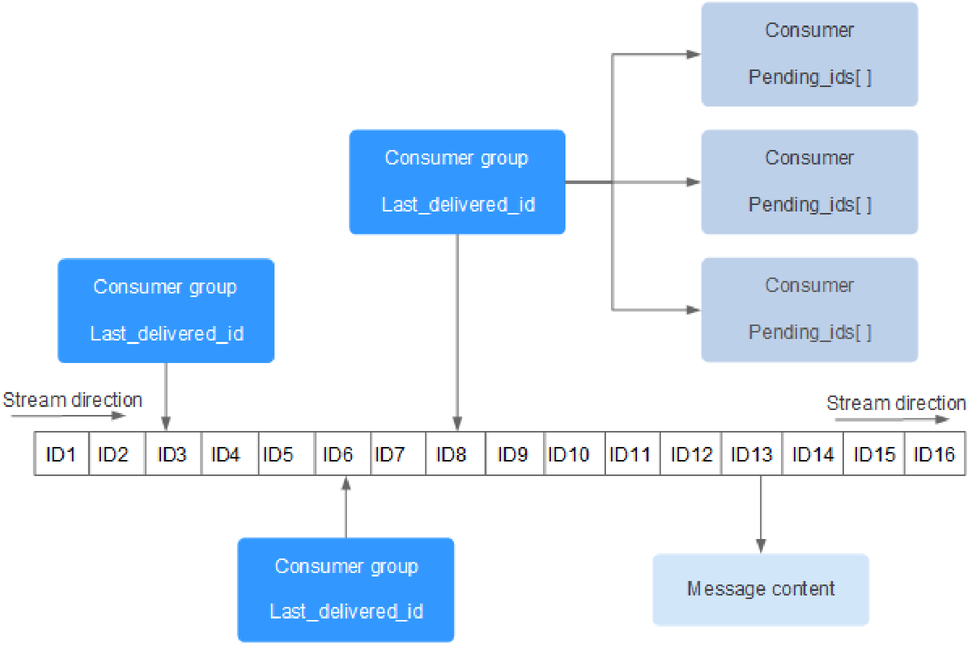


Table 1-4 Differences between streams and existing Redis data structures

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.
Acknowledgement	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

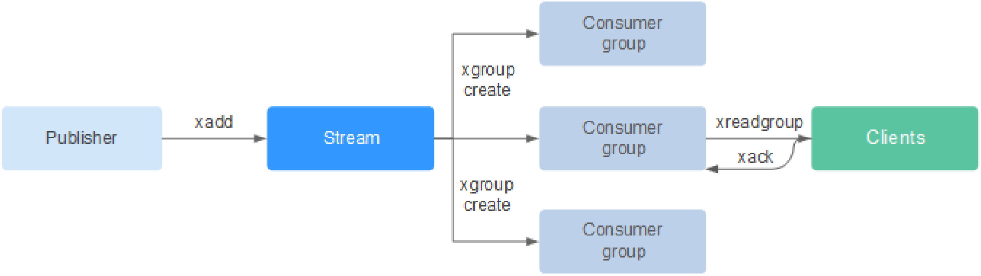


Table 1-5 Stream commands description

Command	Description	Syntax
XACK	Deletes one or multiple messages from the *pending entry list* (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: ● 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 *last delivery ID* 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	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

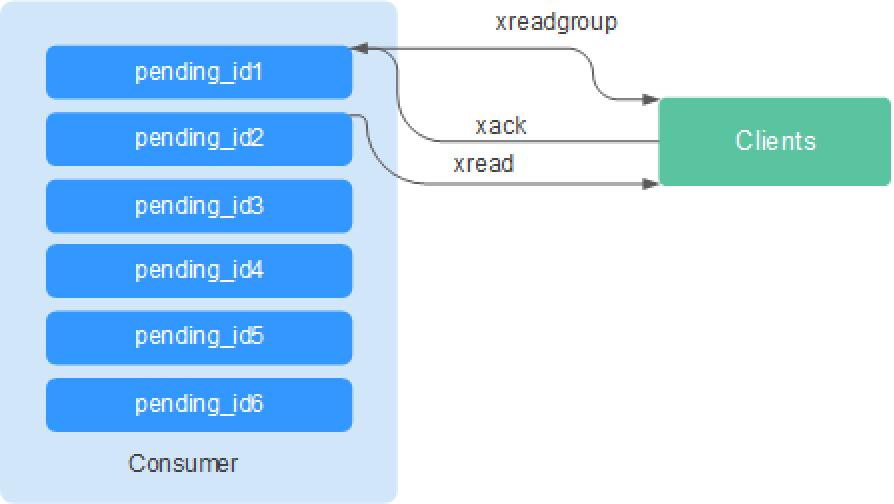
Message (stream item) acknowledgement

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 Acknowledgement 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**

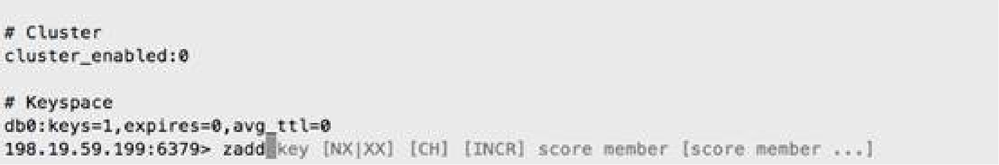
   The **help** command can be used to view help information, saving you the trouble of visiting **redis.io** every time. For example, run the following command to view the stream help information: **xinfo help**

```
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 How Do I Choose Between Arm and x86?

DCS for Redis fully supports both Arm and x86 CPU architectures. They do not differ in functions or client compatibility.

However, Kunpeng and x86 differ in the following aspects:

- Supported Redis versions
  - Arm: Redis 4.0 and Redis 5.0
  - x86: Redis 3.0, Redis 4.0, and Redis 5.0

- Supported instance types
  - Arm: Single-node, master/standby, and Redis Cluster
  - x86: Single-node, master/standby, Redis 3.0 Proxy Cluster, and Redis 4.0 or 5.0 Redis Cluster
- Prices

  Kunpeng-based Redis is 30% cheaper than x86-based Redis.
- Performance

  Performance of different instance specifications is listed in **DCS Instance Specifications**.

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).

In conclusion, both Arm-based Redis and x86-based Redis provide performance that is capable of meeting your service requirements, but Arm-based Redis is more cost-effective.

2 Client and Network Connection

2.1 How Do I Configure a Security Group?

The following describes how to configure security groups for **intra-VPC access** and **public access**.

Intra-VPC Access to DCS Instances

An ECS can communicate with a DCS instance if they belong to the same VPC and same subnet and security group rules are configured correctly.

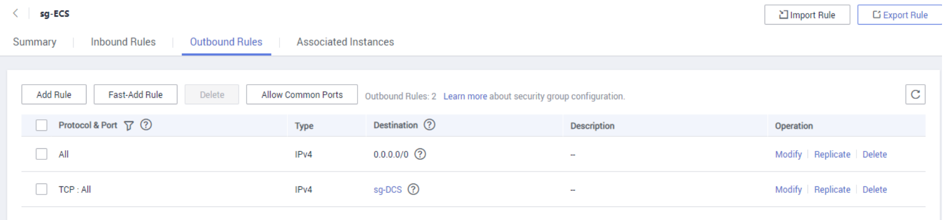
- If the ECS and DCS instance are configured with the same security group, network access in the group is not restricted by default.
- If the ECS and DCS instance are configured with different security groups, add security group rules to ensure that the ECS and DCS instance can access each other.

  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**.
  - Suppose that the port number of the DCS service is 6379.
  - The remote end is a security group or an IP address.

  a. Configuring security group for the ECS.

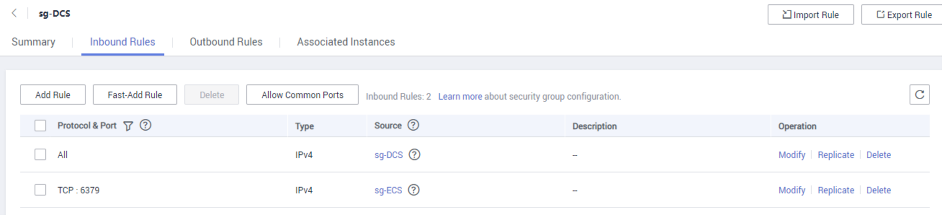
     Add the following outbound rule to allow the ECS to access the DCS instance.



□ Protocol & Port	Type	Destination	Description	Operation
□ All	IPv4	0.0.0.0/0	--	Modify | Replicate | Delete
□ TCP : All	IPv4	sg-DCS	--	Modify | Replicate | Delete

  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:



Protocol & Port	Type	Source	Description	Operation
All	IPv4	sg-DCS	--	Modify | Replicate | Delete
TCP : 6379	IPv4	sg-ECS	--	Modify | Replicate | Delete

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.

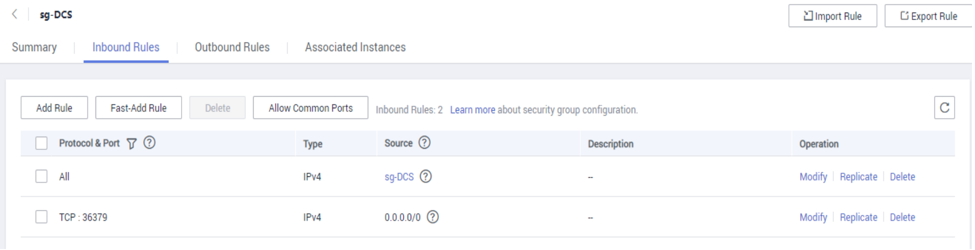
Public Access to DCS Instances

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:

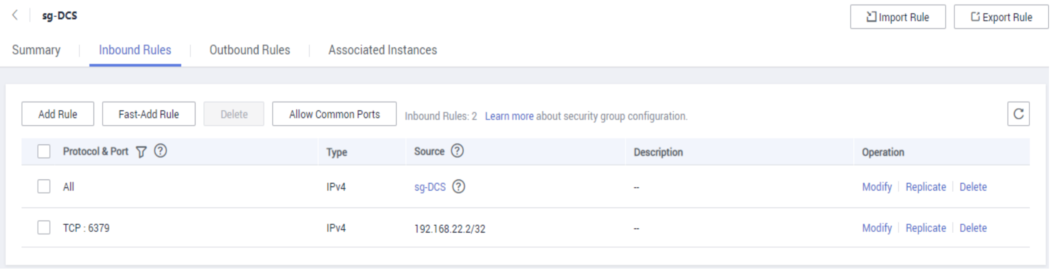
Protocol: **TCP**; source IP address: **0.0.0.0/0** or a specified client address. **When SSL is enabled, set the port number to 36379. When SSL is disabled, set the port number to 6379**.

Figure 2-1 Security group rule (port 36379)



Protocol & Port	Type	Source	Description	Operation
All	IPv4	sg-DCS	--	Modify | Replicate | Delete
TCP : 36379	IPv4	0.0.0.0/0	--	Modify | Replicate | Delete

Figure 2-2 Security group rule (port 6379)



Protocol & Port	Type	Source	Description	Operation
All	IPv4	sg-DCS	--	Modify | Replicate | Delete
TCP : 6379	IPv4	192.168.22.2/32	--	Modify | Replicate | Delete

2.2 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 5.0

  Public access is not supported by DCS Redis 4.0 and 5.0 instances. If public access is required, 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 to Access DCS Redis 4.0 or 5.0 Instances over Public Networks**.

- 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 phase, you can also use an SSH agent to access DCS instances in the local environment.

2.3 Troubleshooting Redis Connection Exceptions

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**
- **Password**
- **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 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 or 5.0 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

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**: 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**.

Client Connections

- If the connection fails when you use redis-cli to connect to a Redis Cluster instance, do as follows:

  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**.

- If error "Read timed out" or "Could not get a resource from the pool" occurs, do as follows:

  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 avoid executing the command frequently.

- 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**.

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**.

2.4 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.

However, by establishing VPC peering connections between VPCs, an ECS can access a 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**.

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

Yes.

Before enabling public access, you must have an available EIP. For the billing details, see the **EIP pricing details**.

2.6 How Can I Access DCS Across Sub-projects?

Enable public access for the instance.

2.7 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.

2.8 Why Does a Redis Cluster DCS Instance Fail to Be Pinged?

The **PING** command is disabled on the node where the instance resides.

2.9 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.

2.10 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.

If a connection request times out, check if AOF persistence has been enabled. To avoid blocking, ensure that AOF has been enabled.

If timeout errors occur frequently, contact technical support.

2.11 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 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 belong to the same VPC and security group, or the security group of your DCS instance allows access through port 6379. For details, see **Security Group Configurations**.

- 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 the Jedis connection pool. 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

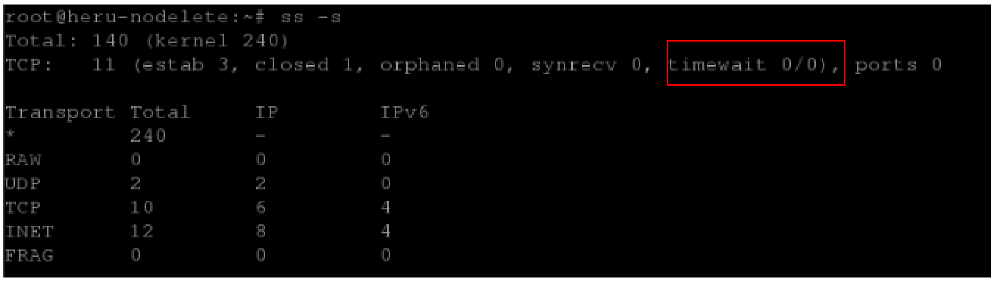
Step 3 Check the JedisPool code.

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.

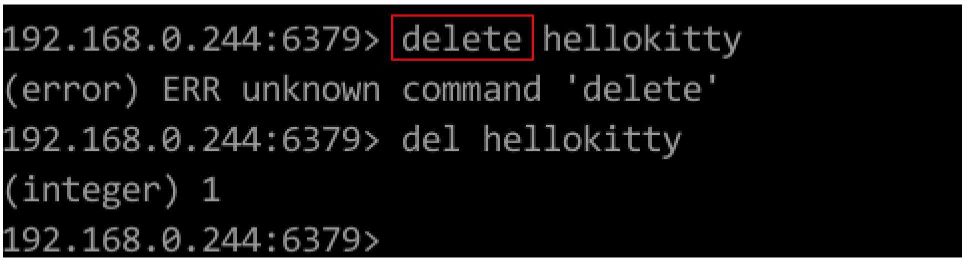
----End

2.12 Why Is "ERR unknown command" Displayed When I Access a DCS Redis Instance Through a Redis Client?

The possible causes are as follows:

1. The command is spelled incorrectly.

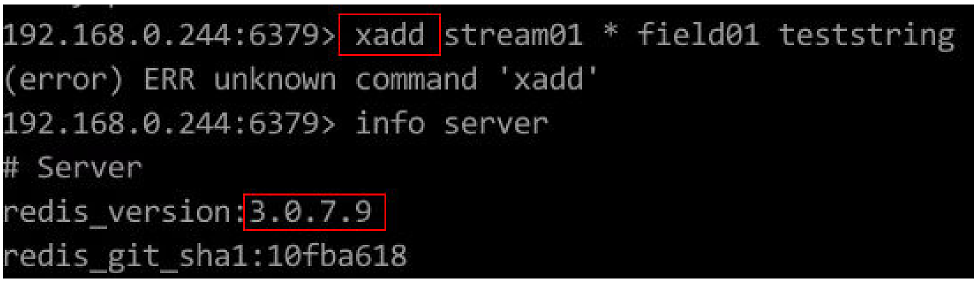
   As shown in the following figure, the error message is returned because the correct command for deleting a string should be **del**.



```
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>
```

2. 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
```

3. Some commands are disabled.

   DCS Redis instance interfaces are fully compatible with the open-source Redis in terms of data access. However, for ease of use and security purposes, some operations cannot be initiated through Redis clients. For details about disabled commands, see **Redis Command Compatibility**.

4. 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

5. The **EVAL** command is now compatible with recent single-node and master/standby DCS Redis instances. Older instances still do not support the command, and "ERR unknown command" will be displayed if **EVAL** is run. You can upgrade older instances by submitting a service ticket. Beware that during the upgrade, the Redis processes will be restarted and the services will be interrupted.

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

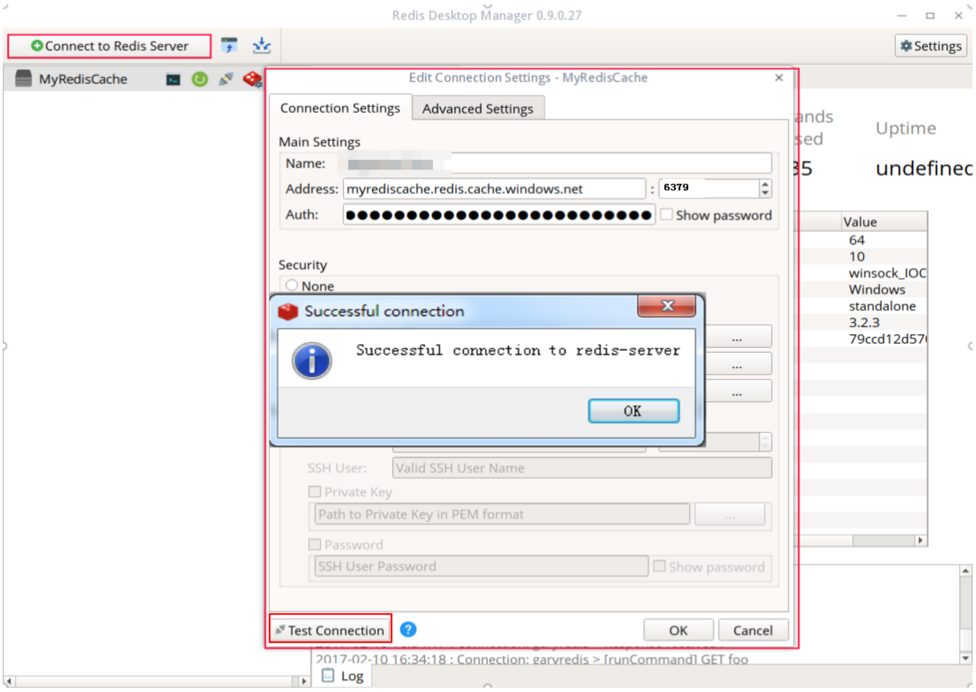
You can access a DCS Redis instance through the Redis Desktop Manager within a VPC or over the Internet.

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 2-3 Accessing a DCS Redis instance through Redis Desktop Manager over the intranet



Main Settings

Name: [redacted]

Address: myrediscache.redis.cache.windows.net : 6379

Auth: ●●●●●●●●●●●●●●●●●●●●●●●●● □ Show password

Security

○ None

SSH User: Valid SSH User Name

□ Private Key

Path to Private Key in PEM format

□ Password

SSH User Password □ Show password

Successful connection

Successful connection to redis-server

OK

64
10
winsock_IOC
Windows
standalone
3.2.3
79ccd12d57

```
2017-02-10 16:34:18 : Connection: garvredis > [runCommand] GET foo
```

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.

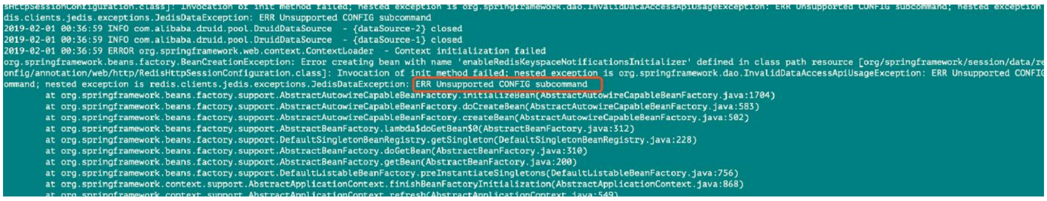
    Configure the inbound rule of the security group of the instance, allowing access over port 36379.

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.

2.14 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 2-4 Spring Cloud error information



```
sHttpSessionConfiguration.class]: Invocation of init method failed; nested exception is org.springframework.dao.InvalidDataAccessApiUsageException: ERR Unsupported CONFIG subcommand; nested exception
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/re
onfig/annotation/web/http/RedisHttpSessionConfiguration.class]: Invocation of init method failed; nested exception is org.springframework.dao.InvalidDataAccessApiUsageException: ERR Unsupported CONFIG
ommand; nested exception is redis.clients.jedis.exceptions.JedisDataException: ERR Unsupported CONFIG subcommand
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1704)
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:583)
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:502)
        at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:312)
        at org.springframework.beans.factory.support.DefaultSingletonBeanRegistry.getSingleton(DefaultSingletonBeanRegistry.java:228)
        at org.springframework.beans.factory.support.AbstractBeanFactory.doGetBean(AbstractBeanFactory.java:310)
        at org.springframework.beans.factory.support.AbstractBeanFactory.getBean(AbstractBeanFactory.java:200)
        at org.springframework.beans.factory.support.DefaultListableBeanFactory.preInstantiateSingletons(DefaultListableBeanFactory.java:756)
        at org.springframework.context.support.AbstractApplicationContext.finishBeanFactoryInitialization(AbstractApplicationContext.java:868)
        at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:549)
```

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.

2.15 Why Do 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?**

2.16 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.

2.17 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 that is in the same VPC as the instance. VPCs are used to ensure network security of public cloud 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.
- If public access is enabled, DCS instances can be accessed in local environments. For more information, see **Public Access to a DCS Redis Instance**.

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

Pay attention to the following issues when using Redis for pub/sub:

- 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.

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

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

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

3 Redis Usage

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

- Possible cause 1:

  The service QPS is so high that the CPU usage spikes to 100%.

- Possible cause 2:

  You have run commands that consume a lot of resources, such as **KEYS**. This will make CPU usage spike and can easily trigger a master/standby switchover.

  In this case, use the **SCAN** command instead or disable the **KEYS** command.

- Possible cause 3:

  The persistence function has been enabled for the instance. If the persistence function is not required, disable it by going to the **Parameters** tab page of the instance and changing the value of **appendonly** to **no**.

3.2 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**.

3.3 Why Can't Security Groups Configured for DCS Redis 4.0 and 5.0 Instances?

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

To restrict access to DCS Reids 4.0 and 5.0 instances from specified IP addresses, you can **manage the IP address whitelist**.

3.4 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.

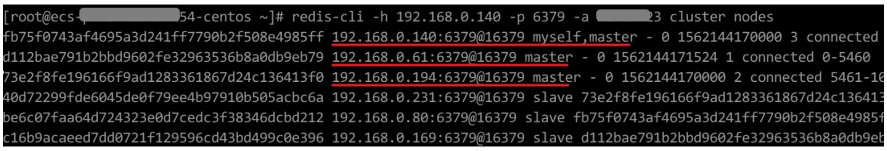
3.5 Can I Obtain the Addresses of the Nodes in a Cluster DCS Redis Instance?

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 5.0 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-[redacted]54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a [redacted]23 cluster nodes
fb75f0743af4695a3d241ff7790b2f508e4985ff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-1
40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
c16b9acaeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
```

3.6 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 does not occur in DCS Redis 4.0 and 5.0 instances.

3.7 Does DCS for Redis Support Read/Write Splitting?

Currently, read/write splitting is supported only by Redis Cluster DCS instances and master/standby DCS Redis 4.0 or 5.0 instances.

- For Redis Cluster instances, you can query all master and replica nodes by running the **CLUSTER NODES** command. The client will select the right node to write or read data.

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

- For master/standby DCS Redis 4.0 and 5.0 instances, you can query the read/write domain name address (master node) and read/only domain name address (standby node) on the instance basic information page.

3.8 Does DCS for Redis Support Multi-DB?

Both single-node and master/standby DCS Redis instances support multiple databases (multi-DB). By default, single-node and master/standby DCS instances can read and write data in 256 databases (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 DB may fully occupy the memory of the instance.

Cluster DCS instances do not support multi-DB.

The DB quantity cannot be modified.

3.9 Does DCS for Redis Support Redis Clusters?

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

3.10 Does DCS for Redis Support Sentinel?

Yes. Redis Sentinel is supported by DCS for Redis 4.0 and 5.0 and is enabled by default. Sentinel constantly checks if master and replica nodes are running properly. If the master is not running properly, Sentinel starts a failover process and promotes a replica to master.

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

3.11 What Is the Default Data Eviction Policy?

Data is evicted from cache based on a user-defined space limit in order to make space for new data. In the current versions of DCS for Redis, you can select an eviction policy you prefer.

noeviction is the default eviction policy for single-node and master/standby DCS Redis instances. You can change the eviction policy by configuring the instance parameters on the DCS console.

volatile-lru is the default eviction policy for cluster DCS Redis instances. To change the eviction policy for cluster instances, submit a service ticket.

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

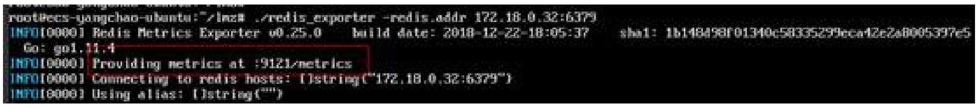
- **noeviction**: When the memory limit is reached, DCS instances return errors to clients and no longer process write requests and other requests that could result in more memory to be used. However, **DEL** and a few more exception requests can continue to be processed.
- **allkeys-lru**: DCS instances 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.

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**.

3.12 What Should I Do If an Error Occurs in Redis Exporter?

Start the Redis exporter using the CLI. Based on the output, check for errors and troubleshoot accordingly.



```
root@ecs-yangchao-ubuntu:~/lmz# ./redis_exporter -redis.addr 172.18.0.32:6379
INFO[0000] Redis Metrics Exporter v0.25.0    build date: 2018-12-22-18:05:37    sha1: 1b148498f01340c58335299eca42e2a8005397e5
  Go: go1.11.4
INFO[0000] Providing metrics at :9121/metrics
INFO[0000] Connecting to redis hosts: []string{"172.18.0.32:6379"}
INFO[0000] Using alias: []string{""}
```

3.13 Why Is Memory Usage More Than 100%?

This is normal due to Redis functions (such as master/replica replication and lazyfree). When the memory becomes full, scale up the instance or remove unnecessary data.

3.14 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 enable 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.

- 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.

3.15 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**, **PEXIPRE**, 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.

3.16 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 or Memcached instance. You can customize but cannot change the port for accessing a DCS Redis 4.0 or 5.0 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 and Redis 5.0

  You can specify a port (ranging from 1 to 65535) or use the default port (6379) for accessing a DCS Redis 4.0 or 5.0 instance. If no port is specified, the default port will be used.

  Public access is not supported.

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?**.

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

After a DCS instance is created, its connection addresses 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 different connection addresses, 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.

3.18 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.

3.19 Does DCS Support Cross-AZ Deployment?

Master/Standby and cluster DCS Redis instances and DCS Memcached instances can be deployed across availability zones (AZs).

- If instances 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.

3.20 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.

3.21 What If Redis Commands Are Incompatible with DCS for Redis?

You can verify whether the commands used by your applications are compatible with DCS by analyzing your service commands or purchase a pay-per-use DCS instance for trial use.

1. DCS for Redis 3.0 has integrated Redis 4.0 commands, which can be used properly in most cases. However, some commands are not supported or are disabled. For details, see **Redis Command Compatibility**.
2. DCS for Redis 3.0 does not support Redis 5.0 commands, such as Stream commands.

3.22 Does DCS for Redis Provide Backend Management Software?

No. To query Redis configurations and usage information, use redis-cli. If you wish to monitor DCS Redis instance metrics, go to the Cloud Eye console. For details on how to configure and view the metrics, see the *Monitoring* chapter in the *DCS User Guide*.

3.23 Why Is Memory of a DCS Redis Instance Used Up by Just a Few Keys?

Possible cause: The output buffer may have occupied an excessive amount of memory.

Solution: After connecting to the instance using redis-cli, run the **redis-cli --bigkeys** command to scan the large key. Then, run the **info** command to check the output buffer size.

3.24 Can I Recover Data from Deleted DCS Instances?

If a DCS instance is automatically deleted or manually deleted through the Redis client, its data cannot be retrieved. If you have backed up the instance, you can restore its data from the backup. However, the restoration will overwrite the data written in during the period from the backup and the restoration.

By default, data is not evicted from DCS instances. You can modify the instance configuration parameters to adjust the eviction policy so that the instance can evict key values.

3.25 Does DCS for Redis Support SSL Encryption?

The open-source Redis does not support SSL-encrypted connections.

However, DCS for Redis 3.0 instances support SSL-encrypted connections if you have enabled public access and **installed the Stunnel client**.

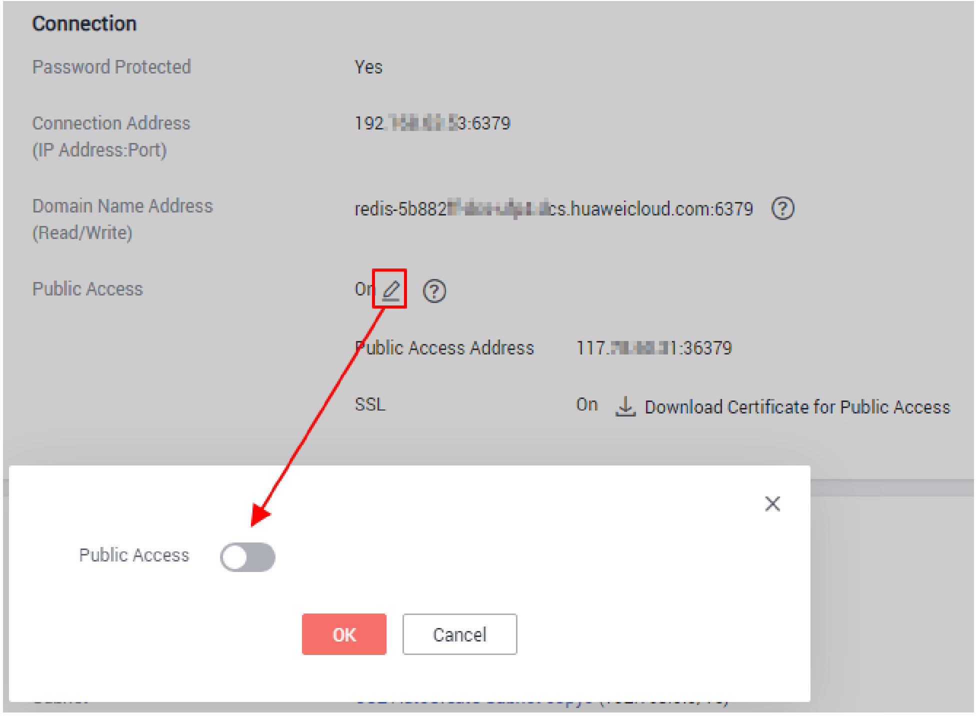
If public access is not enabled, SSL encryption cannot be configured in the Stunnel client.

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

When you enable public access, SSL is enabled by default. **Once an instance is created with public access enabled, the SSL setting cannot be changed**. If you need to disable SSL after the instance has been created, perform the following steps:

1. Disable public access as shown below.

Figure 3-1 Disabling public access



Connection

Password Protected: Yes

Connection Address (IP Address:Port): 192.[redacted]3:6379

Domain Name Address (Read/Write): redis-5b882[redacted]cs.huaweicloud.com:6379

Public Access: On

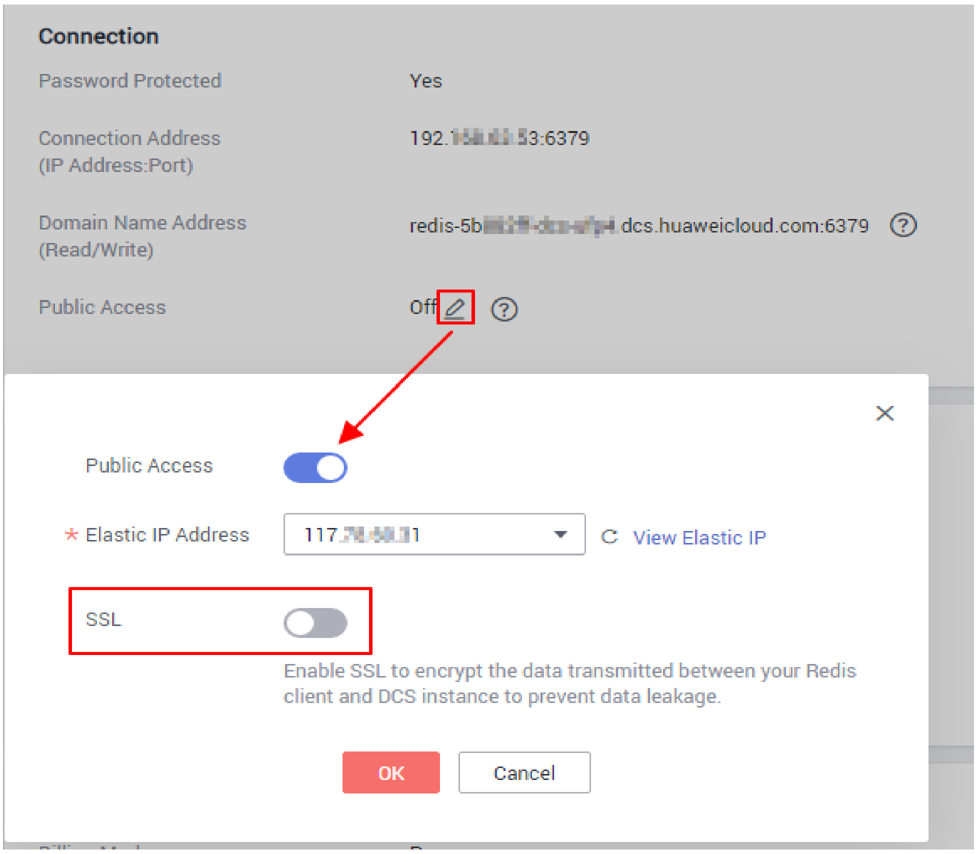
Public Access Address: 117.[redacted]1:36379

SSL: On

Wait until public access has been disabled.

2. Re-enable public access. During this step, disable SSL as shown below.

Figure 3-2 Disabling SSL while enabling public access



Connection

Password Protected: Yes

Connection Address (IP Address:Port): 192.[redacted]53:6379

Domain Name Address (Read/Write): redis-5b[redacted].dcs.huaweicloud.com:6379

Public Access: Off

Public Access: on

* Elastic IP Address: 117.[redacted]1

SSL: off

Enable SSL to encrypt the data transmitted between your Redis client and DCS instance to prevent data leakage.

3.27 Why Is Memory Usage of a DCS Instance 100%?

- Possible cause 1: The expiration time is not set for the instance. As a result, a large amount of data is stored.

  In this case, log in to the DCS console to perform big key analysis on the **Cache Analysis** page and query slow logs on the **Slow Log** page. If the expiration time is not set for the instance, data stored in the instance will use up the memory.
- Possible cause 2: Due to the lazy free feature of Redis, expired keys may stay in the memory.

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

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.

3.29 How Do I Estimate Redis Memory Usage?

The estimated memory usage may be different from the actual memory usage. Currently, DCS for Redis provides the following memory-related metrics:

Table 3-1 DCS Redis 3.0 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: Single-node, master/ standby, or cluster DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory	Used Memory	Number of bytes used by the Redis server Unit: byte	≥ 0	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Dimension: dcs_instance_id	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Raw Data)
used_memory_dataset	Used Memory Dataset	Dataset memory that the Redis server has used Unit: byte	≥ 0	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Supported by Redis 4.0 and later Dimension: dcs_instance_id	1 minute
used_memory_dataset_perc	Used Memory Dataset Ratio	Percentage of dataset memory that the Redis server has used Unit: %	0–100%	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Supported by Redis 4.0 and later Dimension: dcs_instance_id	1 minute
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: byte	≥ 0	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Dimension: dcs_instance_id	1 minute

Metric ID	Metric Name	Description	Value Range	Monitored Object and Dimension	Monitoring Period (Raw Data)
memory_frag_ratio	Memory Fragmentation Ratio	Current memory fragmentation, which is the ratio between **used_memory_rss**/ **used_memory**.	≥ 0	Monitored object: Single-node, master/ standby, or cluster 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: byte	≥ 0	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Dimension: dcs_instance_id	1 minute
used_memory_lua	Used Memory Lua	Number of bytes used by the Lua engine Unit: byte	≥ 0	Monitored object: Single-node, master/ standby, or cluster DCS Redis instance Dimension: dcs_instance_id	1 minute

3.30 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.

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

No. Currently, DCS for Redis does not support external extensions, plug-ins, or modules.

4 Instance Scaling and Upgrade

4.1 Can DCS Redis Instances Be Upgraded, for Example, from Redis 3.0 to Redis 4.0 or 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**.

4.2 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.

4.3 Are Instance Resources Affected During Specification Modification?

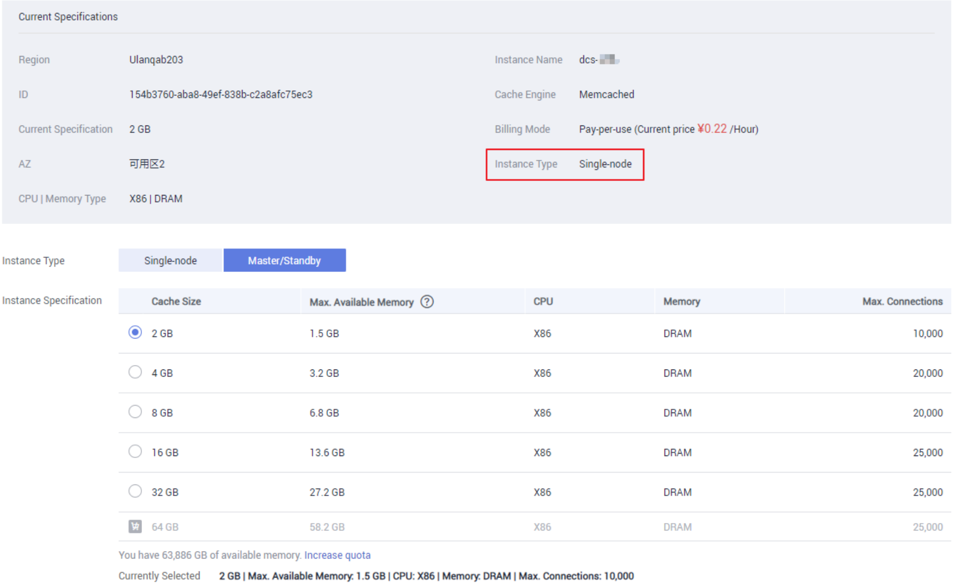
No. Specification modifications can take place while the instance is running and do not affect any other resources.

4.4 Can I Change a Single-Node DCS Redis Instance to the Master/Standby or Cluster Type?

A single-node DCS Redis 3.0 can be changed to the master/standby type. A master/standby DCS Redis 3.0 instance can be changed to the Proxy Cluster type.

The instance type of a DCS Redis 4.0 and 5.0 instance cannot be changed.

To check whether you can change the instance type for an instance, see the parameters displayed on the **Modify Specifications** page on the DCS console. The following scenario shows that the instance type can be changed.



Current Specifications

Region	Ulanqab203	Instance Name	dcs-[redacted]
ID	154b3760-aba8-49ef-838b-c2a8afc75ec3	Cache Engine	Memcached
Current Specification	2 GB	Billing Mode	Pay-per-use (Current price ¥0.22 /Hour)
AZ	可用区2	Instance Type	Single-node
CPU | Memory Type	X86 | DRAM		

Instance Type: Single-node [✓ Master/Standby]

Instance Specification

	Cache Size	Max. Available Memory	CPU	Memory	Max. Connections
●	2 GB	1.5 GB	X86	DRAM	10,000
○	4 GB	3.2 GB	X86	DRAM	20,000
○	8 GB	6.8 GB	X86	DRAM	20,000
○	16 GB	13.6 GB	X86	DRAM	25,000
○	32 GB	27.2 GB	X86	DRAM	25,000
	64 GB	58.2 GB	X86	DRAM	25,000

You have 63,886 GB of available memory. Increase quota

Currently Selected 2 GB | Max. Available Memory: 1.5 GB | CPU: X86 | Memory: DRAM | Max. Connections: 10,000

4.5 Are Services Interrupted During Specification Modification?

You are advised to change the instance specifications during off-peak hours because specification modification has the following impacts:

- **Impact of instance type changes:**
  - From single-node to master/standby:

    The instance cannot be connected for several second and remains read-only for about 1 minute.
  - From master/standby to Proxy Cluster:

    The instance cannot be connected and remains read-only for 5 to 30 minutes.
- **Impact of capacity expansion and reduction:**
  - Single-node and master/standby:

    The instance cannot be connected for several second and remains read-only for about 1 minute.

    For capacity expansion, only the memory of the instance is expanded. The CPU processing capability is not improved.

    Data of single-node instances may not be retained because they do not support data persistence. After the scaling, check whether the data is complete and import data if required.
  - Proxy Cluster:

    The instance can be connected, but the CPU will be occupied and the latency will increase during data migration. During capacity expansion, new Redis Server nodes are added, and data is automatically balanced to the new nodes.

Backup records created before the capacity change cannot be restored.

- Redis Cluster:

  The instance can be connected, but the CPU usage will increase and the latency will increase during data migration. During capacity expansion, new Redis Server nodes are added, and data is automatically balanced to the new nodes.

4.6 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.

4.7 How Do I Reduce the Capacity of a Cluster DCS Instance?

No. Capacity reduction is supported only for single-node and master/standby instances. The capacity of a Proxy Cluster or Redis Cluster instance can only be expanded, and cannot be reduced.

- If you want to use a smaller Proxy Cluster 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**.
- If you want to use a smaller Redis Cluster instance, create a new Redis Cluster instance with the desired capacity, and migrate data online from the old instance to the new instance. After the migration is complete, delete the old instance. For details about data migration operations, see **Migrating Data Online**.

5 Data Backup, Export, and Migration

5.1 How Do I Export DCS Redis Instance Data?

- For master/standby or cluster instances:

  Perform the following operations to export the 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.

- For single-node instances:

  Single-node instances do not support the backup function. You can use redis-cli to export 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.

5.2 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.

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

- Redis 3.0

  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}
- Redis 4.0 and 5.0

  Yes. Backup data of a DCS Redis 4.0 or 5.0 instance is exported from the console to RDB files. You cannot use redis-cli to export such data to RDB files.

5.4 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.

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

The backup files are kept by Object Storage Service (OBS). Currently, DCS backup operations are free of charge, but OBS charges for the storage based on the storage space and duration.

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

Migration between single-node and master/standby instances involves the full set of data. All DBs will be migrated, and you cannot migrate specified DBs. After the migration, a given key will remain in the same DB as it was before the migration.

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

5.7 Does DCS Support Data Persistence?

1. DCS Redis instances:
   - Single-node: Not supported

   - Master/Standby and cluster: Supported
2. DCS Memcached instances:
   - Single-node: Not supported
   - Master/Standby: Supported

5.8 Why Does Data Migration Fail?

- 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.
- To migrate data from a single-node or master/standby instance to a Proxy Cluster instance, check if any data exists in databases other than DB0 of the source instance. If yes, move the data to DB0. Otherwise, the migration fails.

  A Proxy Cluster only has one DB, which is DB0.

6 Analysis of Big Keys and Hot Keys

6.1 What Is the Impact of a Big Key?

A big key occupies a large amount of bandwidth and memory resources, affecting other key operations.

6.2 What Is the Impact of a Hot Key?

A hot key occupies a large amount of CPU and bandwidth resources, affecting other key operations.

6.3 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.

6.4 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.

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

- 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.

The preceding method for discovering hot keys can be used for DCS Redis 3.0 instances, because hot key analysis is not supported by DCS Redis 3.0 instances.

For details about setting alarm rules, see **Setting Alarm Rules for Critical Metrics**.

For details about the operations of big key and hot key analysis, see **Analyzing Big Keys and Hot Keys**.

7 Redis Commands

7.1 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 DB.

- Redis 4.0 and 5.0

  Data of a DCS Redis 4.0 or 5.0 instance can be cleared all at once on the console or by the **FLUSHDB** or **FLUSHALL** command in redis-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.

7.2 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 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**.

7.3 Why Do I Fail to Execute Some Redis Commands?

Possible causes include the following:

- The command is incorrect.
- 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 **Redis 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.

7.4 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.

7.5 How Do I Rename High-Risk Commands?

Currently, you can only rename the following critical commands: **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, and **HGETALL** for DCS Redis 4.0 and 5.0 instances.

These commands can be renamed during instance creation or by using the command renaming function on the console after the instance is created. The renaming takes effect after the instance is restarted.

7.6 Does DCS for Redis Support Pipelining?

Yes. For DCS Redis 4.0 and 5.0 instances in the Redis Cluster mode, ensure that all commands in a pipeline are executed on the same shard.

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

Yes. For more information about Redis command compatibility, see **Redis Command Compatibility**.

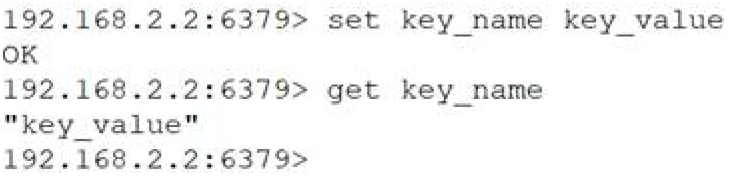
7.8 Why Does a Redis Command Fail to Take Effect?

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.

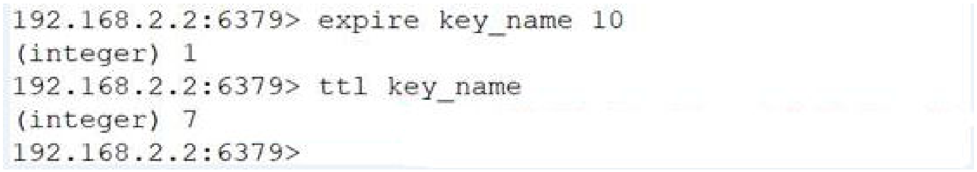
  The **SET** command is used to set the string value. If the value is not changed, run the following commands in redis-cli to access the instance:

```
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.

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

The time limit for executing a Redis command is 1 minute. This limit cannot be configured. After the execution of a command times out, your client will be automatically disconnected.

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

No. Like in open-source Redis, keys in DCS for Redis are case-sensitive.

7.11 Can I View the Most Frequently Used Redis Commands?

No. Redis does not record commands and does not support viewing top commands.

8 Monitoring and Alarm

8.1 How Do I View Concurrent Connections of a DCS Redis Instance?

The number of connections received by a DCS instance is a metric that can be monitored by Cloud Eye. 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.

On the Cloud Eye console, you can also view other monitoring metrics of your DCS instances, for example:

- CPU Usage
- Memory Usage
- Used Memory
- Ops per Second

8.2 Does Redis Support Command Audits?

No. To ensure high-performance reads and writes, Redis does not audit commands. Commands are not printed.

8.3 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 **http://www.redis.io/commands/info**.

8.4 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.

Why is used_memory higher than max_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.

8.5 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: Master/standby DCS Redis 4.0 or 5.0 instances Redis Server of Redis Cluster DCS Redis 4.0 or 5.0 instance Dimension: dcs_cluster_node	1 minute

Bandwidth usage = (Input flow + Output flow)/(2 x Maximum bandwidth) x 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 traffic is greater than the normal service traffic.

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.

8.6 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.

9 Master/Standby Switchover

9.1 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.

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.

9.2 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.

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

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

9.4 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.

10 Purchasing and Billing

10.1 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 resources are sold out.

  Analysis: The underlying server resources might be insufficient to fulfill instance creation orders submitted by multiple users at the same time. Therefore, some users might fail to create instances.

  Solution: Delete the instance failed to be created and create a new one in another AZ.

- 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.

10.2 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**.

11 Memcached Usage

11.1 Can I Dump DCS Memcached Instance Data for Analysis?

No.

11.2 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.

11.3 What Data Structures Does DCS for Memcached Support?

Only the key-value structure is supported.

11.4 Does DCS for Memcached Support Public Access?

No. 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 phase, you can also use an SSH agent to access DCS instances in the local environment. For details, see **Can I Access DCS Instances in a Local Environment?**

11.5 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**.

11.6 What Are the Differences Between DCS for Memcached and Self-Hosted Memcached?

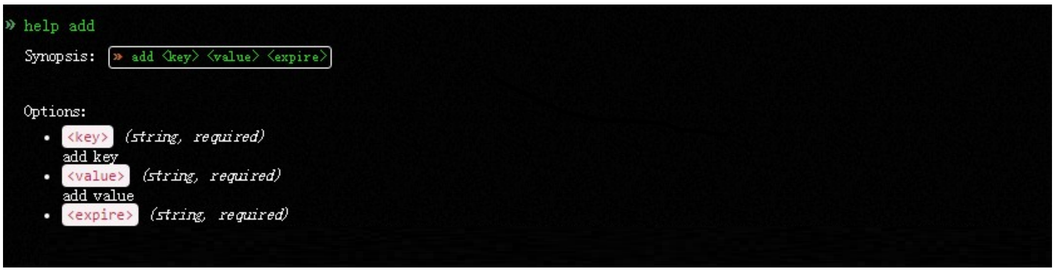
Table 11-1 describes the differences between DCS for Memcached and self-hosted Memcached.

Table 11-1 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.

11.7 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 Is the Default Data Eviction Policy?**

11.8 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 belongs to **AZ B** in the **CN South-Guangzhou** region. When purchasing a DCS Memcached instance, you can select any AZ in **CN South-Guangzhou**. If you select **AZ B** in **CN South-Guangzhou**, 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.

12 DCS Password Complexity Requirements

Passwords for DCS instances must meet the following requirements:

- Cannot be left blank.
- Cannot be the username or the username spelled backwards.
- Must be 8 to 32 characters long.
- Contain at least three of the following character types:
  - Lowercase letters
  - Uppercase letters
  - Digits
  - Special characters (`~!@#$^&*()-_=+\|{}:,<.>/?)