Distributed Cache Service

Service Overview

Issue 01

Date 2025-07-25





Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2025. All rights reserved.

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

Trademarks and Permissions

HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road

Qianzhong Avenue Gui'an New District Gui Zhou 550029

People's Republic of China

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

i

Contents

1 Infographics for Comparing DCS for Redis with Open-Source Re	edis1
2 What Is DCS?	3
3 Application Scenarios	8
4 Security	11
4.1 Shared Responsibilities	11
4.2 Identity Authentication and Access Control	13
4.3 Data Protection	13
4.4 Audit and Logs	14
4.5 Resilience	14
4.6 Security Risks Monitoring	15
4.7 Certificates	15
4.8 Security White Paper	17
5 Selecting DCS Products	19
6 DCS Instance Types	22
6.1 Single-Node Redis	22
6.2 Master/Standby Redis	24
6.3 Proxy Cluster Redis	28
6.4 Redis Cluster	33
6.5 Read/Write Splitting Redis	35
6.6 Comparing DCS Redis Instance Types	38
6.7 Single-Node Memcached (Discontinued)	
6.8 Master/Standby Memcached (Discontinued)	44
7 DCS Instance Specifications	46
7.1 Redis 4.0/5.0/6.0/7.0	46
7.2 Redis 6.0 Enterprise Edition	64
7.3 Redis 3.0 Instance Specifications (Discontinued)	65
7.4 Memcached Instance Specifications (Discontinued)	68
8 Command Compatibility	71
8.1 Commands Supported and Disabled by DCS for Redis 4.0	71
8.2 Commands Supported and Disabled by DCS for Redis 5.0	81

8.3 Commands Supported and Disabled by DCS for Redis 6.0	92
8.4 Commands Supported and Disabled in Web CLI	108
8.5 Command Restrictions	111
8.6 Other Command Usage Restrictions	118
8.7 Commands Supported and Disabled by DCS for Redis 3.0 (Discontinued)	120
8.8 Commands Supported and Disabled by DCS for Memcached (Discontinued)	124
9 Disaster Recovery and Multi-Active Solution	130
10 Comparing Versions and Specifications	135
10.1 Comparing Redis Instance Types	135
10.2 Open Source Redis Versions	137
10.3 Comparing Enterprise and Basic Editions	139
11 Comparing DCS and Open-Source Cache Services	141
12 Notes and Constraints	145
13 Billing	148
14 Permissions Management	150
15 Basic Concepts	155
16 Related Services	157

Infographics for Comparing DCS for Redis with Open-Source Redis



2 What Is DCS?

Huawei Cloud Distributed Cache Service (DCS) is an online, distributed, fast inmemory cache service compatible with Redis. It is reliable, scalable, usable out of the box, and easy to manage, meeting your requirements for high read/write performance and fast data access.

Usability out of the box

DCS provides single-node, master/standby, read/write splitting, Proxy Cluster, and Redis Cluster instances with specifications ranging from 128 MB to 2048 GB. DCS instances can be created with just a few clicks on the console, without requiring you to prepare servers.

Basic edition Redis instances are in container-based deployment in seconds.

Security and reliability

Instance data storage and access are securely protected through HUAWEI CLOUD security management services, including Identity and Access Management (IAM), Virtual Private Cloud (VPC), Cloud Eye, and Cloud Trace Service (CTS).

Master/Standby and cluster instances can be deployed within an availability zone (AZ) or across AZs.

Auto scaling

DCS instances can be scaled up or down online, helping you control costs based on service requirements.

• Easy management

A web-based console is provided for you to perform various operations, such as restarting instances, modifying configuration parameters, and backing up and restoring data. RESTful application programming interfaces (APIs) are also provided for automatic instance management.

• Online migration

You can create a data migration task on the console to import backup files or migrate data online.

DCS for Redis

DCS for Redis supports Redis 4.0, 5.0, 6.0, and 7.0.

■ NOTE

DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 5.0 or later instead.

Redis is a storage system that supports multiple types of data structures, including key-value pairs. It can be used in such scenarios as data caching, event publishing/subscribing, and high-speed queuing, as described in **Application Scenarios**. Redis is written in ANSI C, supporting direct read/write of **strings**, **hashes**, **lists**, **sets**, **sorted sets**, and **streams**. Redis works with an in-memory dataset which can be persisted on disk.

DCS Redis instances can be customized based on your requirements.

• DCS for Redis 4.0/5.0/6.0/7.0 basic edition

Table 2-1 DCS for Redis 4.0/5.0/6.0/7.0 basic edition

Instance type	DCS for Redis provides the following types of instances to suit different service scenarios:	
	 Single-node: Suitable for caching temporary data in low reliability scenarios. Single-node instances support highly concurrent read/write operations, but do not support data persistence. Data will be deleted after instances are restarted. 	
	 Master/standby: Each master/standby instance runs on two nodes (one master and one standby). The standby node replicates data synchronously from the master node. If the master node fails, the standby node automatically becomes the master node. You can split read and write operations by writing to the master node and reading from the standby node. This improves the overall cache read/write performance. 	
	 Proxy Cluster: In addition to the native Redis cluster, a Proxy Cluster instance has proxies and load balancers. Load balancers implement load balancing. Different requests are distributed to different proxies to achieve high-concurrency. Each shard in the cluster has a master node and a standby node. If the master node is faulty, the standby node on the same shard is promoted to the master role to take over services. 	
	 Redis Cluster: Each Redis Cluster instance consists of multiple shards and each shard includes a master node and multiple replicas (or no replica at all). Shards are not visible to you. If the master node fails, a replica on the same shard takes over services. You can split read and write operations by writing to the master node and reading from the replicas. This improves the overall cache read/write performance. 	
	 Read/write splitting: A read/write splitting instance has proxies and load balancers in addition to the master/standby architecture. Load balancers implement load balancing, and different requests are distributed to different proxies. Proxies distinguish between read and write requests, and sends them to master nodes or standby nodes, respectively. 	
Instance specificat ions	DCS for Redis provides instances of different specifications, ranging from 128 MB to 1,024 GB.	
Open- source compatib ility	DCS instances are compatible with open-source Redis 4.0/5.0/6.0/7.0.	
Instance performa nce	QPS per shard for a x86-based Redis instance: 100,000/s; QPS per shard for an Arm-based Redis instance: 80,000/s.	

For more information about open-source Redis, visit https://redis.io/.

• DCS for Redis 6.0 enterprise edition

Huawei Cloud DCS enterprise edition is completely developed by Huawei Cloud. This edition uses the multi-thread master-N*worker model instead of the conventional master-worker model, improving overall performance for n times. DCS enterprise edition is fully compatible with Redis engines, modules, and scripts in terms of scripts and event atomicity. Using the same hardware, this edition doubles the QPS of Redis, and reduces latency by around 60%.

In versions earlier than Redis 6.0, a slow query often causes other queries to be delayed due to the single thread model. To address performance issues, the new edition has made a series of optimization based on a multi-thread model. Multi-thread concurrency has been improved for I/O and backend event processing; access to cached data is further accelerated through fair spinlock; expired keys can be removed twice as faster thanks to optimized algorithms; support for subkey expires also improves the read/write performance of big keys. As a result, the new edition is suitable in scenarios requiring high single-node performance, such as trending topics and livestreaming events on the Internet.

Table 2-2 DCS for Redis 6.0 enterprise edition

Instance type	DCS for Redis 6.0 enterprise edition includes performance and storage sub-editions. Only master/standby instances are available. The enterprise (storage) edition uses memory and SSD disks.
	Each master/standby instance runs on two nodes (one master and one standby). The standby node replicates data synchronously from the master node. If the master node fails, the standby node automatically becomes the master node. You can split read and write operations by writing to the master node and reading from the standby node. This improves the overall cache read/write performance.
Instance specificat ions	8 GB, 16 GB, 32 GB, 64 GB
Open- source compatib ility	Fully compatible with Redis 6.

Instance performa nce	QPS per shard for an enterprise (performance) instance node: 400,000/s; QPS per shard for an enterprise (storage) instance node: 70,000/s.
HA and DR	All instances except single-node ones can be deployed across AZs within a region with physically isolated power supplies and networks.

DCS for Memcached (Discontinued)

DCS for Memcached is no longer provided. You can use DCS Redis instances instead.

Memcached is an in-memory key-value caching system that supports read/write of simple strings. It is often used to cache backend database data to alleviate load on these databases and accelerate web applications. For details about its application scenarios, see Memcached (Discontinued) Application Scenarios.

In addition to full compatibility with Memcached, DCS for Memcached provides the hot standby and data persistence.

Table 2-3 DCS Memcached instance configuration

Instance type	DCS for Memcached provides the following two types of instances to suit different service scenarios:
	Single-node: Suitable for caching temporary data in low reliability scenarios. Single-node instances support highly concurrent read/write operations, but do not support data persistence. Data will be deleted after instances are restarted.
	Master/Standby: Each master/standby instance runs on two nodes (one master and one standby). The standby node replicates data synchronously from the master node, but does not support read/write operations. If the master node fails, the standby node automatically becomes the master node.
Memory	Specification of single-node or master/standby DCS Memcached instances: 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, and 64 GB.
HA and DR	Master/Standby DCS Memcached instances can be deployed across AZs in the same region with physically isolated power supplies and networks.

For more information about open-source Memcached, visit https://memcached.org/.

3 Application Scenarios

Redis Application Scenarios

Many large-scale e-commerce websites and video streaming and gaming applications require fast access to large amounts of data that has simple data structures and does not need frequent join queries. In such scenarios, you can use Redis to achieve fast yet inexpensive access to data. Redis enables you to retrieve data from in-memory data stores instead of relying entirely on slower disk-based databases. In addition, you no longer need to perform additional management tasks. These features make Redis an important supplement to traditional disk-based databases and a basic service essential for Internet applications receiving high-concurrency access.

Typical application scenarios of DCS for Redis are as follows:

1. E-commerce flash sales

E-commerce product catalogue, deals, and flash sales data can be cached to Redis.

For example, the high-concurrency data access in flash sales can be hardly handled by traditional relational databases. It requires the hardware to have higher configuration such as disk I/O. By contrast, Redis supports 100,000 QPS per node and allows you to implement locking using simple commands such as **SET**, **GET**, **DEL**, and **RPUSH** to handle flash sales.

For details about locking, see **Implementing Distributed Locks with Redis** in *Best Practices*.

2. Live video commenting

In live streaming, online user, gift ranking, and bullet comment data can be stored as sorted sets in Redis.

For example, bullet comments can be returned using the **ZREVRANGEBYSCORE** command. The **ZPOPMAX** and **ZPOPMIN** commands in Redis 5.0 can further facilitate message processing.

3. Game leaderboard

In online gaming, the top 10 ranking players are displayed and updated in real time. The leaderboard ranking can be stored as sorted sets, which are easy to use with up to 20 commands.

For details, see **Ranking with Redis** in *Best Practices*.

4. Social networking comments

In web applications, queries of post comments often involve sorting by time in descending order. As comments pile up, sorting becomes less efficient.

By using lists in Redis, a preset number of comments can be returned from the cache, rather than from disk, easing the load off the database and accelerating application responses.

Memcached (Discontinued) Application Scenarios

Memcached is suitable for storing simple key-value data.

1. Web pages

Caching static data such as HTML pages, Cascading Style Sheets (CSS), and images to DCS Memcached instances improves access performance of web pages.

2. Frontend database

In dynamic systems such as social networking and blogging sites, write operations are far fewer than read operations such as querying users, friends, and articles. To ease the database load and improve performance, the following data can be cached to Memcached:

 Frequently accessed data that does not require real-time updates and can expire automatically

Example: latest article lists and rankings. Although data is generated constantly, its impact on user experience is limited. Such data can be cached for a preset period of time and accessed from the database after this period. If web page editors want to view the latest ranking, a cache clearing or refreshing policy can be configured.

Frequently accessed data that requires real-time updates
 Example: friend lists, article lists, and reading records. Such data can be cached to Memcached first, and then updated whenever changes (adding, modifying, and deleting data) occur.

3. Flash sales

It is difficult for traditional databases to write an order placement operation during flash sales into the database, modify the inventory data, and ensure transaction consistency while ensuring uninterrupted user experience.

Memcached **incr** and **decr** commands can be used to store inventory information and complete order placement in memory. Once an order is submitted, an order number is generated. Then, the order can be paid.

■ NOTE

Scenarios where Memcached is not suitable:

- The size of a single cache object is larger than 1 MB.
 Memcached cannot cache an object larger than 1 MB. In such cases, use Redis.
- The key contains more than 250 characters.
 - To use Memcached in such a scenario, you can generate an MD5 hash for the key and cache the hash instead.
- High data reliability is required.
 - Open-source Memcached does not provide data replication, backup, and migration, so data persistence is not supported.
 - Master/Standby DCS Memcached instances support data persistence. For more information, contact technical support.
- Complex data structures and processing are required.
 - Memcached supports only simple key-value pairs, and does not support complex data structures such as lists and sets, or complex operations such as sorting.

4 Security

4.1 Shared Responsibilities

Huawei guarantees that its commitment to cyber security will never be outweighed by the consideration of commercial interests. To cope with emerging cloud security challenges and pervasive cloud security threats and attacks, Huawei Cloud builds a comprehensive cloud service security assurance system for different regions and industries based on Huawei's unique software and hardware advantages, laws, regulations, industry standards, and security ecosystem.

Unlike traditional on-premises data centers, cloud computing separates operators from users. This approach not only enhances flexibility and control for users but also greatly reduces their operational workload. For this reason, cloud security cannot be fully ensured by one party. Cloud security requires joint efforts of Huawei Cloud and you, as shown in Figure 4-1.

- Huawei Cloud: Huawei Cloud is responsible for infrastructure security, including security and compliance, regardless of cloud service categories. The infrastructure consists of physical data centers, which house compute, storage, and network resources, virtualization platforms, and cloud services Huawei Cloud provides for you. In PaaS and SaaS scenarios, Huawei Cloud is responsible for security settings, vulnerability remediation, security controls, and detecting any intrusions into the network where your services or Huawei Cloud components are deployed.
- Customer: As our customer, your ownership of and control over your data assets will not be transferred under any cloud service category. Without your explicit authorization, Huawei Cloud will not use or monetize your data, but you are responsible for protecting your data and managing identities and access. This includes ensuring the legal compliance of your data on the cloud, using secure credentials (such as strong passwords and multi-factor authentication), and properly managing those credentials, as well as monitoring and managing content security, looking out for abnormal account behavior, and responding to it, when discovered, in a timely manner.

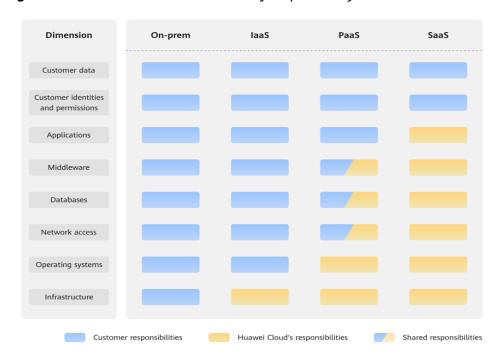


Figure 4-1 Huawei Cloud shared security responsibility model

Cloud security responsibilities are determined by control, visibility, and availability. When you migrate services to the cloud, assets, such as devices, hardware, software, media, VMs, OSs, and data, are controlled by both you and Huawei Cloud. This means that your responsibilities depend on the cloud services you select. As shown in **Figure 4-1**, customers can select different cloud service types (such as IaaS, PaaS, and SaaS services) based on their service requirements. As control over components varies across different cloud service categories, the responsibilities are shared differently.

- In on-premises scenarios, customers have full control over assets such as hardware, software, and data, so tenants are responsible for the security of all components.
- In IaaS scenarios, customers have control over all components except the underlying infrastructure. So, customers are responsible for securing these components. This includes ensuring the legal compliance of the applications, maintaining development and design security, and managing vulnerability remediation, configuration security, and security controls for related components such as middleware, databases, and operating systems.
- In PaaS scenarios, customers are responsible for the applications they deploy, as well as the security settings and policies of the middleware, database, and network access under their control.
- In SaaS scenarios, customers have control over their content, accounts, and permissions. They need to protect their content, and properly configure and protect their accounts and permissions in compliance with laws and regulations.

4.2 Identity Authentication and Access Control

Identity Authentication

Access requesters must present the identity credential for identity validity verification when accessing DCS on the console or by calling APIs. DCS uses Identity and Access Management (IAM) to provide three identity authentication modes: passwords, access keys, and temporary access keys. In addition, DCS provides login protection and login authentication policies to harden identity authentication security.

Access Control

You can assign different permissions for DCS to employees in your organization for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your Huawei Cloud resources. For details, see **Permissions Management**.

4.3 Data Protection

DCS takes different measures to keep data secure and reliable.

Table 4-1 DCS data protection methods and features

Measure	Description	Reference
DR and multi- active	To meet reliability requirements of your data and services, you can deploy a DCS instance in a single AZ (single equipment room) or across AZs (intracity DR).	Disaster Recovery and Multi-Active Solution
Data replication	Data can be incrementally synchronized between replicas for cache data consistency. When a network exception or node fault occurs, a failover is automatically performed using the replicas. After the fault is rectified, a full synchronization is performed to ensure data consistency.	Data Replication
Data persistence	Exceptions may occur during daily running of the service system. Some service systems require high reliability, including high availability of instances, cache data security, recoverability, and even permanent storage, so that backup data can be used to restore instances if an exception occurs, ensuring service running.	Backing Up and Restoring Instances

4.4 Audit and Logs

Cloud Trace Service (CTS)

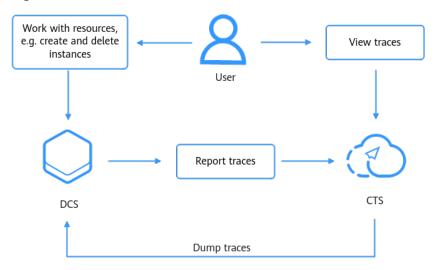
CTS records operations on the cloud resources in your account. You can use the logs generated by CTS to perform security analysis, trace resource changes, audit compliance, and locate faults.

After you enable CTS and configure a tracker, CTS can record management and data traces of DCS for auditing.

For details about how to enable and configure CTS, see **Enabling CTS**.

For details about DCS management and data operations that can be traced, see **Operations Logged by CTS**.

Figure 4-2 CTS



4.5 Resilience

DCS provides a three-level reliability architecture and uses cross-AZ DR, intra-AZ instance DR, and instance data replication to ensure service durability and reliability.

Table 4-2 DCS reliability architecture

Reliability Solution	Description
Cross-AZ DR	DCS provides master/standby, Redis Cluster, and Proxy Cluster instances that support cross-AZ DR. When an AZ is abnormal, the instances can still provide services.

Reliability Solution	Description
Intra-AZ instance DR	A master/standby, Redis Cluster, or Proxy Cluster DCS instance has multiple nodes for instance-level DR within an AZ. If the master node is faulty, services are quickly switched to a standby node to ensure uninterrupted DCS services.
Data DR	Data DR is implemented through data replication.

4.6 Security Risks Monitoring

DCS uses Cloud Eye to help you monitor your DCS instances and receive alarms and notifications in real time. You can obtain key information about instances in real time, such as service requests, resource usage, bandwidth, number of concurrent operations, and flow control times.

For details about DCS metrics and how to create alarm rules, see DCS Metrics.

4.7 Certificates

Compliance Certificates

Huawei Cloud services and platforms have obtained various security and compliance certifications from authoritative organizations, such as International Organization for Standardization (ISO). You can **download** them from the console.

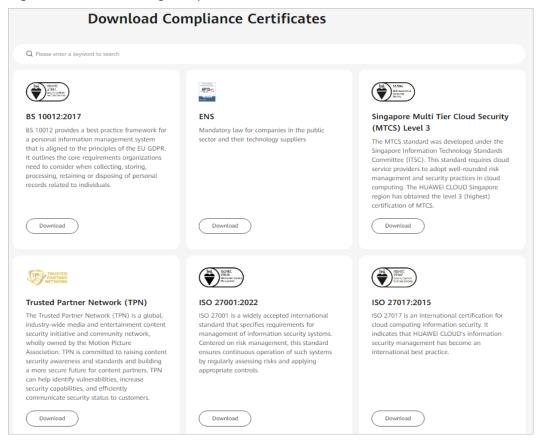
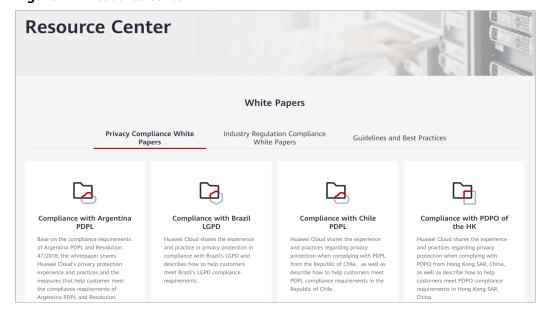


Figure 4-3 Downloading compliance certificates

Resource Center

Huawei Cloud also provides the following resources to help users meet compliance requirements. For details, see **Resource Center**.





4.8 Security White Paper

Distributed Cache Service (DCS) is a secure and reliable in-memory database service provided by Huawei Cloud.

DCS complies with security regulations, adheres to service boundaries, and will never monetize customer data. It allows you to quickly provision different types of instances and supports auto scaling of compute and storage resources as required. To prevent data loss, DCS provides functions such as automated backups, snapshots, and restorations. It also allows you to modify configuration parameters for instance tuning.

DCS provides many features to ensure the reliability and security of account data, including VPCs, security groups, whitelists, SSL encryption for public access, automated backups, data snapshot, and cross-AZ deployment.

For details about how DCS ensures data security, see DCS Security Best Practices.

Network Isolation

You can configure VPC inbound rules to allow specific IP address segments to connect to your instances. DCS instances run in an independent VPC. You can create a cross-AZ subnet group and deploy high-availability instances in it. After an instance is created, DCS will assign a subnet IP address to the instance for connection. After DCS instances are deployed in a VPC, you can use a VPN to access the instances from other VPCs. You can also create an ECS in the VPC housing the instances and connect the ECS and instances through a floating IP address. Subnets and security groups can be used together to isolate DCS instances and enhance security.

Access Control

When creating a DCS instance, you can configure security groups (supported by DCS for Redis 3.0, Redis 6.0 professional edition, and Memcached) or whitelists (supported by DCS for Redis 4.0, 5.0, and 6.0 basic edition).

You can set inbound and outbound security group rules or configure whitelists to control the access to and from DCS instances within a VPC.

You do not need to restart instances when configuring security groups or whitelists.

When creating a DCS instance, you are advised to enable password protection and set an access password for the instance to prevent unauthenticated clients from accessing the instance by mistake.

Transmission and Storage Encryption

RESP (REdis Serialization Protocol), the communication protocol of Reids, only supports plaintext transmission in versions earlier than Redis 6.0. DCS Redis 6.0 basic edition instances support the RESP3 protocol and encryption over SSL.

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

If you need to encrypt data in transit when public access is not enabled, use an encryption algorithm (such as AES 256) to encrypt data before storage and keep access in the trusted domain. The data is also encrypted before persistence to disk.

Automated and Manual Backups

DCS instances can be backed up automatically or manually. The automated backup function is disabled by default. Backup data of an instance is stored for a maximum of 7 days. After automated backup is enabled, you can restore data to the instance. During automated backup, all data of an instance is backed up, and the performance of the standby node will be affected. Manual backups are user-initiated full backups of instances. The backup data is stored in OBS buckets and removed upon deletion of the corresponding instance.

Data Replication

A master/standby or cluster DCS instances can be deployed within an AZ or across multiple AZs for HA. For cross-AZ deployment, DCS initiates and maintains data synchronization. High availability is achieved by having a standby node take over in the event that a failure occurs on the master node. When operations are readheavy, you can use DCS Redis 4.0 or later instances that support read/write splitting, or cluster instances that have multiple replicas. DCS maintains data synchronization between the master and replicas. You can connect to different addresses of an instance to isolate read and write operations.

Data Deletion

If you delete a DCS instance, all data stored in the instance will be deleted. Nobody can view or restore the data once it is deleted.

Selecting DCS Products

Select a DCS Redis instance based on your service scenarios, performance requirements, and budget. This document provides references for selecting product editions, instance types, and instance specifications.

Selecting a Product Edition

Table 5-1 Product editions

Preference	Suggestion	Analysis
Performance and enterprise- class capabilities	Enterprise (Performance) (Enterprise (Memory))	 Performance: multithreading and multi-core scheduling, higher performance at a single node, and more stable access latency in the case of high concurrency Enterprise-level capabilities: data flashback Rich modules: JSON and Bitmap
Balanced performance and cost	Basic edition	 Performance: slightly higher than that of open-source self-built Redis, but lower than that of the enterprise (memory) edition. High availability: Master/Standby replication and automatic failover are available. High elasticity: Vertical and horizontal scaling satisfies capacity and performance requirements.
Cost- effectiveness	Enterprise (storage) edition (Enterprise (capacity) edition)	 Performance: equivalent to that of open-source self-built Redis. Low cost: With cold/hot separation, the SSD stores full data and the memory stores hot data. Therefore, the cost is the lowest and the performance is basically the same when the capacity is the same.

Preference	Suggestion	Analysis
Ultra-large bandwidth	Basic edition Redis Cluster instance	Assured bandwidth: Compared with Proxy Cluster instances whose maximum bandwidth is 10,000 Mbit/s, bandwidth of a Redis Cluster instance can be increased as the number of shards increases.
		Elastic bandwidth: On each shard, the bandwidth can be increased to 2,048 Mbit/s.

□ NOTE

- Enterprise (performance) and enterprise (storage) editions are in upgrades. They will become enterprise (memory) and enterprise (capacity) editions. To protect customer rights and reduce O&M complexity, enterprise features will be available later. Enterprise edition instances are currently in restricted sales.
- For details about the differences between DCS editions, see Comparing Redis Instance Types.

Selecting an Instance Type

Table 5-2 Instance types and replica quantities

Category	Description	Scenarios	
Single-node	Single replica (one node). Data persistence is unavailable. The instance is inaccessible and data will be lost when the node is faulty.	Only for development and test scenarios. Do not use it in a production environment. There is no Service Level Agreement (SLA) assurance.	
Master/ Standby	Two replicas (one master node and one standby node). Data persistence is available. Node faults can be rectified through master/standby switchovers, ensuring access.	Production services whose capacity is small, performance is stable at the 100,000 QPS level, and expansion is not required.	
Proxy Cluster	Two replicas (one master node and one standby node) by default. Forward proxies shield backend network topology changes, reducing client access complexity.	Production services whose data capacity is large and concurrency is in the tens of millions.	
Redis Cluster	Starting with two replicas (one master node and one standby node) is recommended. Full compatibility with opensource Redis.	Production services whose data capacity is large and concurrency is in the tens of millions.	

Category	Description	Scenarios
Read/Write splitting	At least three replicas (one master node and at least two standby nodes). Based on the architecture of a master/ standby instance, forward proxies schedule read and write requests to the master and standby nodes respectively.	Production services whose data capacity is small, write performance is at the 100,000 QPS level, and read performance is at the 100,000s QPS level.

Selecting an Instance Flavor

DCS Instance Specifications compares the instance specifications supported by different instance types. An instance flavor can be estimated with read/write concurrency, data volume, and maximum bandwidth. Reserve at least 20% more than the estimated requirement can reduce complexity in O&M changes later.

The following provides an example of estimating the flavor of a basic edition instance:

- When read/write performance is at the 100,000 QPS level, select master/ standby.
- When write performance is at the 100,000 QPS level and read concurrency is at the 100,000s QPS level, read/write splitting is recommended.
- When read and write performance is at the 100,000s QPS level, Proxy Cluster or Redis Cluster is required. Select a shard quantity based on performance requirements.
- Reserve at least 20% more than the required specifications.

6 DCS Instance Types

6.1 Single-Node Redis

This section describes single-node DCS Redis instances.

NOTICE

Single-node instances only have one master node, and can be vulnerable of data reliability and service level agreement (SLA) when a physical server is faulty. Exercise caution. You are not advised to use them in production environments. *DCS Service Level Agreement* does not apply to single-node instances.

Features

- Low-cost and suitable for development and testing
 Single-node instances are 40% cheaper than master/standby DCS instances, suitable for setting up development or testing environments.
- Data not persisted
 Data persistence is not ensured through one master node for single-node instances. The data cannot be backed up.

Architecture

Figure 6-1 shows the architecture of a single-node DCS Redis instance.

- Redis 3.0 and 6.0 enterprise do not support port customization and allow only port 6379. For Redis 4.0 and later basic, you can specify a port or use the default port 6379. In the following architecture, port 6379 is used. If you have customized a port, replace **6379** with the actual port.
- You cannot upgrade the Redis version for an instance. For example, a single-node DCS Redis 4.0 instance cannot be upgraded to a single-node DCS Redis 5.0 instance. If your service requires the features of higher Redis versions, create a DCS Redis instance of a higher version and then migrate data from the old instance to the new one.

VPC Customer application xxxx.dcs.huaweicloud.com:6379 DCS instance Detect service DCS server

Figure 6-1 Single-node DCS Redis instance architecture

Architecture description:

VPC

The VPC where all nodes of the instance run.

Application

The client of the instance, which is the application running on an Elastic Cloud Server (ECS).

DCS Redis instances are compatible with Redis, and can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see instance access instructions.

DCS instance

A single-node DCS instance, which has only one node and runs one Redis process.

6.2 Master/Standby Redis

This section describes master/standby DCS Redis instances.

□ NOTE

You cannot upgrade the Redis version for an instance. For example, a master/standby DCS Redis 4.0 instance cannot be upgraded to a master/standby DCS Redis 5.0 instance. If your service requires the features of higher Redis versions, create a DCS Redis instance of a higher version and then migrate data from the old instance to the new one.

Features

Master/Standby DCS instances have higher availability and reliability than single-node DCS instances.

Master/Standby DCS instances have the following features:

1. Data persistence and high reliability

By default, data persistence is enabled by both the master and the standby nodes of a master/standby DCS Redis instance.

The standby node of a Redis 3.0 instance is invisible to you. Only the master node provides data read/write operations.

The standby node of a master/standby Redis 4.0 or later basic edition instance is visible to you. You can read data from the standby node by connecting to it using the instance read-only address.

2. Data synchronization

Data in the master and standby nodes is kept consistent through incremental synchronization.

◯ NOTE

After recovering from a network exception or node fault, master/standby instances perform a full synchronization to ensure data consistency.

3. Automatic master/standby switchover

If the master node becomes faulty, the instance is disconnected and unavailable for several seconds. The standby node takes over for 15 to 30 seconds without manual operations to resume stable services.

- Disconnections and unavailability occur during the failover. The service client should be able to reconnect or retry.
- After a master/standby failover is complete, the previous faulty master node (then
 the standby one) will be recovered later. Service access to the previous master
 node will fail. In this case, configure Redis SDKs. For details, see Connecting to
 Redis on a Client.

4. Multiple DR policies

Each master/standby DCS instance can be deployed across AZs with physically isolated power supplies and networks. Applications can also be deployed across AZs to achieve HA for both data and applications.

5. Read/write splitting

Master/Standby DCS Redis 4.0 and later basic instances support client read/write splitting. When connecting to such an instance, you can use the read/write address to connect to the master node or use the read-only address to connect to the standby node.

If you use a master/standby instance and need client-side read/write splitting, configure the client. If read/write splitting is required, read/write splitting instances are recommended.

Architecture of Master/Standby DCS Redis 3.0 Instances

Figure 6-2 shows the architecture of a master/standby DCS Redis 3.0 instance.

VPC

Application

xxxx.dcs.huaweicloud.com:6379

DCS instance

Replica

Keepalived

Master

Figure 6-2 Architecture of a master/standby DCS Redis 3.0 instance

Architecture description:

VPC

The VPC where all nodes of the instance run.

Application

The Redis client of the instance, which is the application running on the ECS. DCS Redis and Memcached instances are respectively compatible with Redis and Memcached protocols, and can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see the instance access instructions.

DCS instance

A master/standby DCS instance which has a master node and a replica node. By default, data persistence is enabled and data is synchronized between the two nodes

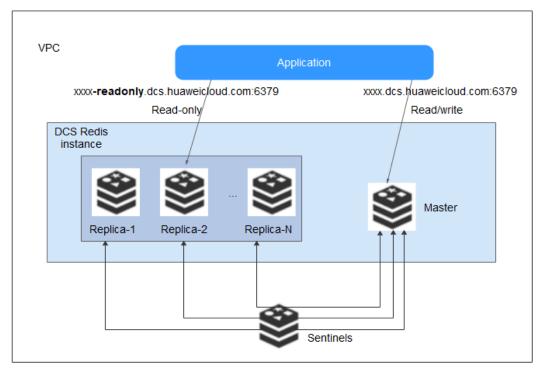
DCS monitors the availability of the instance in real time. If the master node becomes faulty, the standby node becomes the master node and resumes service provisioning.

The default Redis port is 6379.

Architecture of Master/Standby DCS Redis 4.0/5.0/6.0/7.0 Basic Instances

The following figure shows the architecture of a master/standby DCS Redis 4.0/5.0/6.0/7.0 basic instance.

Figure 6-3 Architecture of a master/standby DCS Redis 4.0/5.0/6.0/7.0 basic instance



Architecture description:

- 1. Each master/standby DCS Redis 4.0/5.0/6.0/7.0 basic instance has a domain name address (for connecting to the master node) for read and write and an address (for connecting to the standby node) for read only.
 - These addresses can be obtained on the instance details page on the DCS console.
- 2. You can configure Sentinel for a master/standby Redis 4.0/5.0/6.0/7.0 basic instance. Sentinels monitor the running status of the master and standby nodes. If the master node becomes faulty, a failover will be performed.
 Sentinels are invisible to you and is used only in the service. For details about Sentinel, see What Is Sentinel?
- 3. A read-only node has the same specifications as a read/write node. When a master/standby instance is created, a pair of master and standby nodes are included in the instance by default.

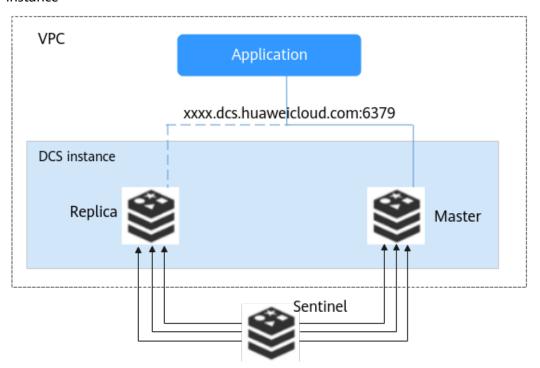
□ NOTE

- For DCS Redis 4.0 and later basic edition instances, you can customize the port. If no port is specified, the default port 6379 will be used. In the architecture diagram, port 6379 is used. If you have customized a port, replace 6379 with the actual port.
- Read-only domain names of master/standby DCS Redis 4.0 and later basic edition instances do not support load balancing. For high reliability and low latency, use cluster or read/write splitting instances.
- Requests to the domain name address may fail if the standby of a master/standby pair (DCS Redis 4.0 or later basic edition) is faulty. For higher reliability and lower latency, use read/write splitting instances.

Architecture of Master/Standby DCS Redis 6.0 Enterprise Edition Instances

Figure 6-4 shows the architecture of a master/standby DCS Redis 6.0 enterprise edition instance.

Figure 6-4 Architecture of a master/standby DCS Redis 6.0 enterprise edition instance



Architecture description:

VPC

The VPC where all nodes of the instance run.

For intra-VPC access, the client and the instance must be in the same VPC with specific security group rule configurations. For details about how to configure the rules, see **How Do I Configure a Security Group?**.

Application

The Redis client of the instance, which is the application running on the ECS.

DCS Redis instances are compatible with Redis, and can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see **Connecting to Redis on a Client**.

DCS instance

A master/standby DCS instance which has a master node and a replica node.

Master/Standby DCS Redis 6.0 enterprise edition instances support Sentinels. Sentinels monitor the running status of the master and standby nodes. If the master node becomes faulty, a failover will be performed.

Sentinels are invisible to you and is used only in the service. For details about Sentinel, see **What Is Sentinel?**

The default Redis port is 6379.

6.3 Proxy Cluster Redis

DCS for Redis provides Proxy Cluster instances, which use Linux Virtual Server (LVS) and proxies to achieve high availability. Proxy Cluster instances have the following features:

- The client is decoupled from the cloud service.
- They support millions of concurrent requests, equivalent to Redis Cluster instances.
- A wide range of memory specifications adapt to different scenarios.

Ⅲ NOTE

- You cannot upgrade the Redis version for an instance. For example, a Proxy Cluster DCS Redis 4.0 instance cannot be upgraded to a Proxy Cluster DCS Redis 5.0 instance. If your service requires the features of higher Redis versions, create a DCS Redis instance of a higher version and then migrate data from the old instance to the new one.
- A Proxy Cluster instance can be connected in the same way that a single-node or master/standby instance is connected, without any special settings on the client. You can use the IP address or domain name of the instance, and do not need to know or use the proxy or shard addresses.

Proxy Cluster DCS Redis 3.0 Instances

Proxy Cluster DCS Redis 3.0 instances are based on x86, compatible with **open source codis** and come with specifications ranging from 64 GB to 1024 GB, meeting requirements for **millions of concurrent connections** and **massive data cache**. Distributed data storage and access is implemented by DCS, without requiring development or maintenance.

Each Proxy Cluster instance consists of load balancers, proxies, cluster managers, and **shards**.

Table 6-1 Total memory, proxies, and shards of Proxy Cluster DCS Redis 3.0 instances

Total Memory	Proxies	Shards
64 GB	3	8

Total Memory	Proxies	Shards
128 GB	6	16
256 GB	8	32
512 GB	16	64
1024 GB	32	128

Figure 6-5 Architecture of a Proxy Cluster DCS Redis 3.0 instance

Architecture description:

VPC

The VPC where all nodes of the instance are run.

◯ NOTE

If public access is not enabled for the instance, ensure that the client and the instance are in the same VPC and configure security group rules for the VPC.

If public access is enabled for the instance, the client can be deployed outside of the VPC to access the instance through the EIP bound to the instance.

For more information, see Public Access to a DCS Redis 3.0 Instance and How Do I Configure a Security Group?

Application

The client used to access the instance.

DCS Redis instances can be accessed using open-source clients. For examples of accessing DCS instances with different programming languages, see **Accessing a DCS Redis Instance**.

LB-M/LB-S

The load balancers, which are deployed in master/standby HA mode. The connection addresses (**IP address:Port** and **Domain Name:Port**) of the cluster DCS Redis instance are the addresses of the load balancers.

Proxy

The proxy server used to achieve high availability and process high-concurrency client requests.

You can connect to a Proxy Cluster instance at the IP addresses of its proxies.

Redis shard

A shard of the cluster.

Each shard consists of a pair of master/replica nodes. If the master node becomes faulty, the replica node automatically takes over cluster services.

If both the master and replica nodes of a shard are faulty, the cluster can still provide services but the data on the faulty shard is inaccessible.

Cluster manager

The cluster configuration managers, which store configurations and partitioning policies of the cluster. You cannot modify the information about the configuration managers.

Proxy Cluster DCS Redis 4.0/5.0/6.0 Basic Instances

Ⅲ NOTE

Proxy Cluster DCS Redis 4.0 and later instances are provided only in some regions.

Proxy Cluster DCS Redis 4.0/5.0/6.0 instances are built based on open-source Redis 4.0/5.0/6.0 and compatible with **open source codis**. They provide multiple large-capacity specifications ranging from 4 GB to 1024 GB.

Table 6-2 lists the number of shards corresponding to different specifications. You can customize the shard size when creating an instance. Currently, the number of replicas cannot be customized. By default, each shard has two replicas.

Memory per shard=Instance specification/Number of shards. For example, if a 48 GB instance has 6 shards, the size of each shard is 48 GB/6 = 8 GB.

Table 6-2 Total memory, proxies, and shards of Proxy Cluster DCS Redis 4.0/5.0/6.0 basic instances

Total Memory	Proxies	Shards	Memory per Shard (GB)
4 GB	3	3	1.33
8 GB	3	3	2.67
16 GB	3	3	5.33
24 GB	3	3	8

Total Memory	Proxies	Shards	Memory per Shard (GB)
32 GB	3	3	10.67
48 GB	6	6	8
64 GB	8	8	8
96 GB	12	12	8
128 GB	16	16	8
192 GB	24	24	8
256 GB	32	32	8
384 GB	48	48	8
512 GB	64	64	8
768 GB	96	96	8
1024 GB	128	128	8

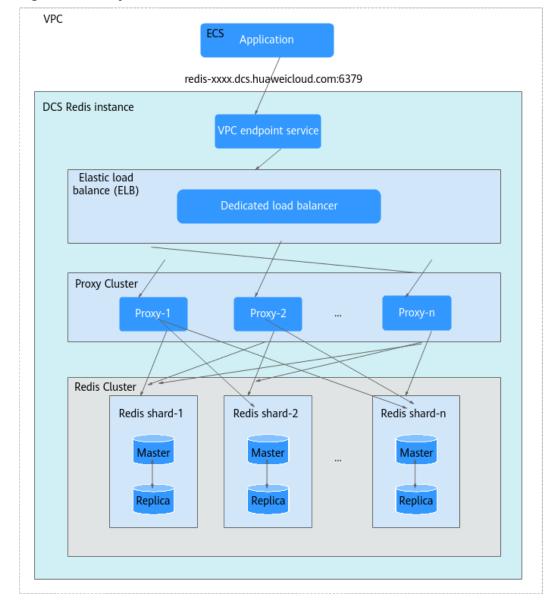


Figure 6-6 Proxy Cluster DCS Redis 4.0/5.0/6.0 basic instances

Architecture description:

VPC

The VPC where all nodes of the instance are run.

Ⅲ NOTE

The client and the cluster instance must be in the same VPC, and the instance whitelist must allow access from the client IP address.

Application

The client used to access the instance.

DCS Redis instances can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see Connecting to Redis on a Client.

• VPC endpoint service

You can configure your DCS Redis instance as a VPC endpoint service and access the instance at the VPC endpoint service address.

The IP address or domain name address of the Proxy Cluster DCS Redis instance is the address of the VPC endpoint service.

ELB

The load balancers are deployed in cluster HA mode and support multi-AZ deployment.

Proxy

The proxy server used to achieve high availability and process high-concurrency client requests.

You cannot connect to a Proxy Cluster instance at the IP addresses of its proxies.

• Redis Cluster

A shard of the cluster.

Each shard is a master/standby dual-replica Redis instance. When the master node is faulty, the standby node will be switched to the master one after 15 to 30 seconds. Access to the shard will fail until the switchover is complete.

If both the master and replica nodes of a shard are faulty, the cluster can still provide services but the data on the faulty shard is inaccessible.

6.4 Redis Cluster

Redis Cluster DCS instances use the native distributed implementation of Redis. Redis Cluster instances have the following features:

- They are compatible with native Redis clusters.
- They inherit the smart client design from Redis.
- They deliver many times higher performance than master/standby instances.

Read/write splitting is supported by configuring the client for Redis Cluster instances. Read more about DCS's support for read/write splitting.

- You cannot upgrade the Redis version for an instance. For example, a Redis Cluster DCS
 Redis 4.0 instance cannot be upgraded to a Redis Cluster DCS Redis 5.0 instance. If your
 service requires the features of higher Redis versions, create a Redis Cluster instance of a
 higher version and then migrate data from the old instance to the new one.
- The method of connecting a client to a Redis Cluster instance is different from that of connecting a client to other types of instances. For details, see Connecting to Redis on a Client.

Architecture

The Redis Cluster instance type provided by DCS is compatible with the **native Redis Cluster**, which uses smart clients and a distributed architecture to perform sharding.

Table 6-3 lists the shard specifications for different instance specifications.

You can customize the shard size when creating a Redis Cluster instance. If the shard size is not customized, the default size is used. **Size of a shard = Instance**

specification/Number of shards. For example, if a 48 GB instance has 6 shards, the size of each shard is 48 GB/6 = 8 GB.

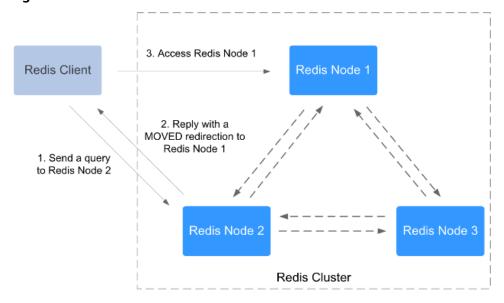
Table 6-3 Specifications of Redis Cluster DCS instances

Total Memory	Shards
4 GB/8 GB/16 GB/24 GB/32 GB	3
48 GB	6
64 GB	8
96 GB	12
128 GB	16
192 GB	24
256 GB	32
384 GB	48
512 GB	64
768 GB	96
1024 GB	128
2048 GB	128

Distributed architecture

Any node in a Redis Cluster can receive requests. Received requests are then redirected to the right node for processing. Each node consists of a subset of one master and one (by default) or multiple replicas. The master or replica roles are determined through an election algorithm.

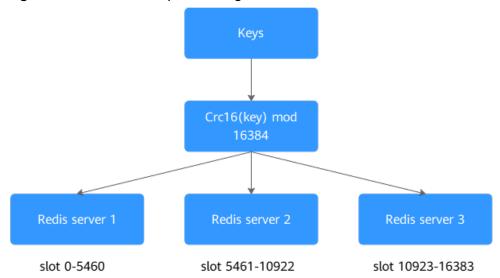
Figure 6-7 Distributed architecture of Redis Cluster



Presharding

There are 16,384 hash slots in each Redis Cluster. The mapping between hash slots and Redis nodes is stored in Redis Servers. To compute what is the hash slot of a given key, simply take the CRC16 of the key modulo 16384.

Figure 6-8 Redis Cluster presharding



◯ NOTE

- Each shard of a Redis Cluster is a master/standby Redis instance. When the master
 node on a shard is faulty, the connections on the shard are interrupted in seconds,
 and the shard becomes unavailable. The standby node is automatically switched
 over within 15 to 30 seconds. The fault only affects the shard itself.
- For master node faults on a single shard, after a master/standby failover is complete, the previous faulty master node (then the standby one) of the shard will be recovered later. Service access to the previous master node of the shard will fail. In this case, configure Redis SDKs. For details, see Connecting to Redis on a Client.

6.5 Read/Write Splitting Redis

Read/write splitting is suitable for scenarios with high read concurrency and few write requests, aiming to improve the performance of high concurrency and reducing O&M costs.

Ⅲ NOTE

- The read/write splitting instance type is supported only in some regions.
- Only DCS Redis 4.0/5.0/6.0 basic edition support read/write splitting instances.
- You cannot upgrade the Redis version for an instance. For example, a read/write splitting DCS Redis 4.0 instance cannot be upgraded to a read/write splitting DCS Redis 5.0 instance. If your service requires the features of higher Redis versions, create a DCS Redis instance of a higher version and then migrate data from the old instance to the new one.

Read/Write splitting is implemented on the server side by default. Proxies distinguish between read and write requests, and forward write requests to the

master node and read requests to the standby node. You do not need to perform any configuration on the client.

When using read/write splitting instances, note the following:

- 1. Read requests are sent to replicas. There is a delay when data is synchronized from the master to the replicas.
 - If your services are sensitive to the delay, do not use read/write splitting instances. Instead, use master/standby or cluster instances.
- Read/write splitting is suitable when there are more read requests than write requests. If there are a lot of write requests, the master and replicas may be disconnected, or the data synchronization between them may fail after the disconnection. As a result, the read performance deteriorates.
 - If your services are write-heavy, use master/standby or cluster instances.
- 3. If a replica is faulty, it takes some time to synchronize all data from the master. During the synchronization, the replica does not provide services, and the read performance of the instance deteriorates.
 - To reduce the impact of the interruption, use an instance with less than 32 GB memory. The smaller the memory, the shorter the time for full data synchronization between the master and replicas, and the smaller the impact of the interruption.

Architecture

VPC **ECS** Application redis-xxxx.dcs.hua/weicloud.com:6379 Read/Write splitting Redis instance VPC endpoint service ELB cluster LB-1 LB-2 Proxy cluster Proxy-2 Proxy-N Proxy-1 Wite Read Master/ Standby instance Write Replica-1 Replica-2 Replica-N Master Sentinel

Figure 6-9 Architecture of a read/write splitting instance

Architecture description:

• VPC endpoint service

You can configure your DCS Redis instance as a VPC endpoint service and access the instance at the VPC endpoint service address.

The IP address or domain name of the read/write splitting DCS Redis instance is the address of the VPC endpoint service.

ELB

The load balancers are deployed in cluster HA mode and support multi-AZ deployment.

Proxy

A proxy cluster is used to distinguish between read requests and write requests, and forward write requests to the master node and read requests to the standby node. You do not need to configure the client.

Sentinel cluster

Sentinels monitor the status of the master and replicas. If the master node is faulty or abnormal, a failover is performed to ensure that services are not interrupted.

• Master/standby instance

A read/write splitting instance is essentially a master/standby instance that consists of a master node and a standby node. By default, data persistence is enabled and data is synchronized between the two nodes.

The master and standby nodes can be deployed in different AZs.

6.6 Comparing DCS Redis Instance Types

Table 6-4 describes the differences between different Redis instance types in terms of features and commands.

DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 5.0 or later.

Table 6-4 Differences between DCS instance types

Item	Single-Node, Read/Write Splitting, or Master/ Standby	Proxy Cluster	Redis Cluster
Redis versio n compa tibility	Single-node and master/ standby: Redis 3.0/4.0/5.0/6.0/ 7.0. Read/Write splitting: Redis 4.0/5.0/6.0.	Redis 3.0/4.0/5.0/6.0 You can select a version when creating an instance.	Redis 4.0/5.0/6.0/7.0 You can select a version when creating an instance.
	The enterprise edition supports only master/standby instances.		

Item	Single-Node, Read/Write Splitting, or Master/ Standby	Proxy Cluster	Redis Cluster
Suppor t	Keyspace notificationsPipelining	 Pipelining, MSET command, and MGET command SCAN command, KEYS command, and Redis Slow Log Pub/Sub 	 Keyspace notifications BRPOP, BLPOP, and BRPOPLPUSH commands Pub/Sub
Restrictions	Single-node instances do not support data persistence, backup, or restoration.	 LUA script is restricted: All keys must be in the same hash slot to avoid errors. Hash tags are recommended. Some commands that contain multiple keys require that the keys must be in the same hash slot to avoid errors. Hash tags are recommended. For details about these multi-key commands, see Multi-Key Commands of Proxy Cluster Instances. Keyspace notifications are not supported. 	 LUA script is restricted: All keys must be in the same hash slot. Hash tags are recommended. The client SDK must support Redis Cluster and be able to process MOVED errors. When you are using pipelining, MSET command, or MGET command, all keys must be in the same hash slot to avoid errors. Hash tags are recommended. When using keyspace notifications, establish connections with every Redis server and process events on each connection. When using a traversing or global command such as SCAN and KEYS, run the command on each Redis server.
Client	Any Redis client	Any Redis client (no need to support the Redis Cluster protocol)	Any client that supports the Redis Cluster protocol

Item	Single-Node, Read/Write Splitting, or Master/ Standby	Proxy Cluster	Redis Cluster
Disabl ed comm ands	Command Compatibility lists disabled commands. Command Restrictions lists the command restricted for read/write splitting instances.	Command Compatibility lists disabled commands. Command Restrictions lists the command restricted for Proxy Cluster instances.	Command Compatibility lists disabled commands. Command Restrictions lists the command restricted for Redis Cluster instances.

Item	Single-Node, Read/Write Splitting, or Master/ Standby	Proxy Cluster	Redis Cluster
Replic	A single-node instance has only one replica. By default, a master/standby or read/write splitting instance has two replicas, with one of them being the master. When creating a master/ standby or read/write splitting DCS Redis instance, you can customize the number of replicas, with one of them being the master. Currently, the number of replicas cannot be customized for DCS Redis 3.0 and enterprise edition Redis 6.0 instances.	Each shard in a cluster has and can only have two replicas, with one of them being the master.	By default, each shard in a cluster has two replicas. The number of replicas on each shard can be customized, with one of them being the master. When creating an instance, you can set the replica quantity to one, indicating that the instance only has the master node. In this case, high data reliability cannot be ensured.

Related FAQs

- Does DCS for Redis Support Multi-DB?
- What Are the CPU Specifications of DCS Instances?
- Does DCS for Redis Support Read/Write Splitting?

6.7 Single-Node Memcached (Discontinued)

NOTE

DCS for Memcached is no longer provided. You can use DCS Redis instances instead.

This section describes the features and architecture of single-node DCS Memcached instances.

Features

- 1. Low system overhead and high QPS
 - Single-node instances do not support data synchronization or data persistence, reducing system overhead and supporting higher concurrency. QPS of single-node DCS Memcached instances reaches up to 100,000.
- 2. Process monitoring and automatic fault recovery
 - With an HA monitoring mechanism, if a single-node DCS instance becomes faulty, a new process is started within 30 seconds to resume service provisioning.
- 3. Out-of-the-box usability and no data persistence
 - Single-node DCS instances can be used out of the box because they do not involve data loading. If your service requires high QPS, you can warm up the data beforehand to avoid strong concurrency impact on the backend database.
- 4. Low-cost and suitable for development and testing
 Single-node instances are 40% cheaper than master/standby DCS instances, suitable for setting up development or testing environments.

In summary, single-node DCS instances support highly concurrent read/write operations, but do not support data persistence. Data will be deleted after instances are restarted. They are suitable for scenarios which do not require data persistence, such as database front-end caching, to accelerate access and ease the concurrency load off the backend. If the desired data does not exist in the cache, requests will go to the database. When restarting the service or the DCS instance, you can pre-generate cache data from the disk database to relieve pressure on the backend during startup.

Architecture

Figure 6-10 shows the architecture of single-node DCS Memcached instances.

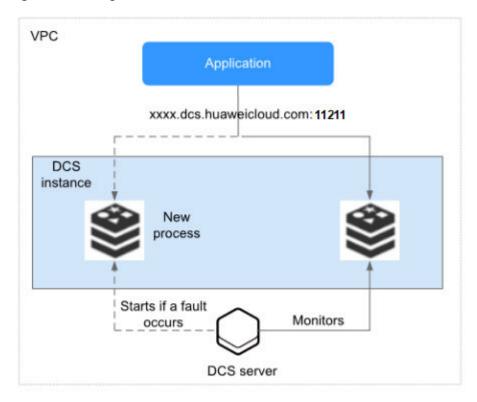


Figure 6-10 Single-node DCS Memcached instance architecture

Architecture description:

VPC

The VPC where all nodes of the instance are run.

□ NOTE

Single-node DCS Memcached instances do not support public access. The client and the instance must be in the same VPC with security group rule configurations.

For more information, see **How Do I Configure a Security Group?**

• Application

The client of the instance, which is the application running on an Elastic Cloud Server (ECS).

DCS Memcached instances are compatible with the Memcached protocol, and can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see Accessing a DCS Memcached Instance.

DCS instance

A single-node DCS instance, which has only one node and one Memcached process.

DCS monitors the availability of the instance in real time. If the Memcached process becomes faulty, DCS starts a new process to resume service provisioning.

Use port 11211 to access a DCS Memcached instance.

6.8 Master/Standby Memcached (Discontinued)

Ⅲ NOTE

DCS for Memcached is no longer provided. You can use DCS Redis instances instead.

This section describes master/standby DCS Memcached instances.

Features

Master/Standby instances have higher availability and reliability than single-node instances.

Master/Standby DCS Memcached instances have the following features:

1. Data persistence and high reliability

By default, data persistence is enabled by both the master and the standby node of a master/standby DCS Memcached instance. In addition, data persistence is supported to ensure high data reliability.

The standby node of a DCS Memcached instance is invisible to you. Only the master node provides data read/write operations.

2. Data synchronization

Data in the master and standby nodes is kept consistent through incremental synchronization.

MOTE

After recovering from a network exception or node fault, master/standby instances perform a full synchronization to ensure data consistency.

3. Automatic master/standby switchover

If the master node becomes faulty, the standby node takes over within 30 seconds, without requiring any service interruptions or manual operations.

4. Multiple DR policies

Each master/standby instance can be deployed across AZs with physically isolated power supplies and networks. Applications can also be deployed across AZs to achieve HA for both data and applications.

Architecture of Master/Standby DCS Memcached Instances

Figure 6-11 shows the architecture of master/standby DCS Memcached instances.

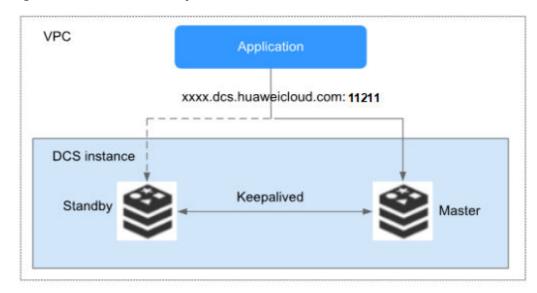


Figure 6-11 Master/Standby DCS Memcached instance architecture

Architecture description:

VPC

The VPC where all nodes of the instance are run.

□ NOTE

Master/Standby DCS Memcached instances do not support public access. The client and the instance must be in the same VPC with security group rule configurations. For more information, see **How Do I Configure a Security Group?**

Application

The Memcached client of the instance, which is the application running on the ECS.

DCS Memcached instances are compatible with the Memcached protocol, and can be accessed through open-source clients. For examples of accessing DCS instances with different programming languages, see **Accessing a DCS**Memcached Instance.

DCS instance

Indicates a master/standby DCS instance which has a master node and a standby node. By default, data persistence is enabled and data is synchronized between the two nodes.

DCS monitors the availability of the instance in real time. If the master node becomes faulty, the standby node becomes the master node and resumes service provisioning.

Use port 11211 to access a DCS Memcached instance.

DCS Instance Specifications

7.1 Redis 4.0/5.0/6.0/7.0

This section describes DCS Redis 4.0/5.0/6.0/7.0 basic edition instance specifications, including the total memory, available memory, maximum number of connections allowed, maximum/assured bandwidth, and reference performance.

The following metrics are related to the instance specifications:

- Used memory: You can check the memory usage of an instance by viewing the **Memory Usage** and **Used Memory** metrics.
- Maximum connections: The maximum number of connections allowed is the
 maximum number of clients that can be connected to an instance. To check
 the number of connections to an instance, view the Connected Clients
 metric. This metric is implemented by the maxclients parameter (unavailable
 for read/write splitting instances). To modify this parameter, go to the
 Instance Configuration > Parameters page on the console.
- QPS represents queries per second, which is the number of commands processed per second. For details about QPS testing, see the Performance White Paper.
- Bandwidth: You can view the Flow Control Times metric to check whether
 the bandwidth has exceeded the limit. You can also check the Bandwidth
 Usage metric. This metric is for reference only, because it may be higher than
 100%. For details, see Why Does Bandwidth Usage Exceed 100%?

□ NOTE

- DCS Redis 4.0, 5.0, and 6.0 basic edition instances are available in single-node, master/standby, Proxy Cluster, Redis Cluster, and read/write splitting types.
- DCS Redis 7.0 instances are available in single-node, master/standby, and Redis Cluster types.
- DCS Redis instances support x86 and Arm architectures. **x86 is recommended. Arm is unavailable in some regions.**
- The specifications available on the console vary by region.
- The Assured/Maximum Bandwidth in the instance specifications is the instance's default bandwidth. To adjust a DCS instance's bandwidth, see Adjusting DCS Instance Bandwidth.

Single-Node Instances

Table 7-1 Specifications of single-node DCS Redis 4.0/5.0/6.0/7.0 instances

Total Memor y (GB)	Available Memory (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Referen ce Perform ance (QPS)	Specification Code (spec_code in the API)
0.125	0.125	10,000/10,0 00	40/40	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.tin y.128 Arm: redis.single.au1.tin y.128
0.25	0.25	10,000/10,0 00	80/80	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.tin y.256 Arm: redis.single.au1.tin y.256
0.5	0.5	10,000/10,0 00	80/80	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.tin y.512 Arm: redis.single.au1.tin y.512
1	1	10,000/50,0 00	80/80	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.1 Arm: redis.single.au1.lar ge.1
2	2	10,000/50,0 00	128/128	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.2 Arm: redis.single.au1.lar ge.2
4	4	10,000/50,0 00	192/192	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.4 Arm: redis.single.au1.lar ge.4

Total Memor y (GB)	Available Memory (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Referen ce Perform ance (QPS)	Specification Code (spec_code in the API)
8	8	10,000/50,0 00	192/192	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.8 Arm: redis.single.au1.lar ge.8
16	16	10,000/50,0 00	256/256	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.16 Arm: redis.single.au1.lar ge.16
24	24	10,000/50,0 00	256/256	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.24 Arm: redis.single.au1.lar ge.24
32	32	10,000/50,0 00	256/256	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.32 Arm: redis.single.au1.lar ge.32
48	48	10,000/50,0 00	256/256	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.48 Arm: redis.single.au1.lar ge.48
64	64	10,000/50,0 00	384/384	x86: 100,000 Arm: 80,000	x86: redis.single.xu1.lar ge.64 Arm: redis.single.au1.lar ge.64

Master/Standby Instances

Each instance has two default replicas (one master node and one standby node) and they can be set.

Given the same memory size, the differences between x86-based master/standby instances, Arm-based master/standby instances, and master/standby instances with multiple replicas are as follows:

- The available memory, maximum number of connections, assured/maximum bandwidth, and QPS are the same.
- Specification code: Table 7-2 only lists the specification names of instances with two default replicas. The specification names reflect the number of replicas, for example, redis.ha.xu1.large.r2.8 (master/standby | x86 | 2 replicas | 8 GB) and redis.ha.xu1.large.r3.8 (master/standby | x86 | 3 replicas | 8 GB).
- IP addresses: Number of occupied IP addresses = Number of master nodes × Number of replicas. For example:
 - 2 replicas: Number of occupied IP addresses = $1 \times 2 = 2$ 3 replicas: Number of occupied IP addresses = $1 \times 3 = 3$

Table 7-2 Specifications of master/standby DCS Redis 4.0/5.0/6.0/7.0 instances

Total Memor y (GB)	Available Memory (GB)	Max. Connect ions (Defaul t/Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performance (QPS)	Specification Code (spec_code in the API)
0.125	0.125	10,000/1 0,000	40/40	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.tiny.r 2.128 Arm: redis.ha.au1.tiny.r 2.128
0.25	0.25	10,000/1 0,000	80/80	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.tiny.r 2.256 Arm: redis.ha.au1.tiny.r 2.256
0.5	0.5	10,000/1 0,000	80/80	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.tiny.r 2.512 Arm: redis.ha.au1.tiny.r 2.512

Total Memor y (GB)	Available Memory (GB)	Max. Connect ions (Defaul t/Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performance (QPS)	Specification Code (spec_code in the API)
1	1	10,000/5 0,000	80/80	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.1 Arm: redis.ha.au1.large. r2.1
2	2	10,000/5 0,000	128/128	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.2 Arm: redis.ha.au1.large. r2.2
4	4	10,000/5 0,000	192/192	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.4 Arm: redis.ha.au1.large. r2.4
8	8	10,000/5 0,000	192/192	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.8 Arm: redis.ha.au1.large. r2.8
16	16	10,000/5 0,000	256/256	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.16 Arm: redis.ha.au1.large. r2.16
24	24	10,000/5 0,000	256/256	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.24 Arm: redis.ha.au1.large. r2.24

Total Memor y (GB)	Available Memory (GB)	Max. Connect ions (Defaul t/Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performance (QPS)	Specification Code (spec_code in the API)
32	32	10,000/5 0,000	256/256	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.32 Arm: redis.ha.au1.large. r2.32
48	48	10,000/5 0,000	256/256	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.48 Arm: redis.ha.au1.large. r2.48
64	64	10,000/5 0,000	384/384	x86: 100,000 Arm: 80,000	x86: redis.ha.xu1.large. r2.64 Arm: redis.ha.au1.large. r2.64

Proxy Cluster Instances

Proxy Cluster instances do not support customization of the number of replicas. Each shard has two replicas by default. For details about the default number of shards, see **Table 6-2**. When buying an instance, you can customize the size of a single shard.

- The following table lists only the Proxy Cluster instance specifications with default shards. If you customize shards, see the maximum number of connections, assured/ maximum bandwidth, and product specification code (flavor) in the Instance Specification table on the Buy DCS Instance page of the DCS console.
- The maximum connections of a cluster is for the entire instance, and not for a single shard. Maximum connections of a single shard = Maximum connections of an instance/ Number of shards.
- The maximum bandwidth and assured bandwidth of a cluster is for the entire instance, and not for a single shard. The relationship between the instance bandwidth and the bandwidth of a single shard is as follows:
 - Instance bandwidth = Bandwidth of a single shard x Number of shards
 - For a cluster instance, if the memory of a single shard is 1 GB, the bandwidth of a single shard is 384 Mbit/s. If the memory of a single shard is greater than 1 GB, the bandwidth of a single shard is 768 Mbit/s.
 - The upper limit of the bandwidth of a Proxy Cluster instance is 10,000 Mbit/s. That is, even if the bandwidth of a single shard multiplied by the number of shards is greater than 10,000 Mbit/s, the bandwidth of the instance is still 10,000 Mbit/s.

Table 7-3 Specifications of Proxy Cluster DCS Redis 4.0/5.0/6.0 instances

Specif icatio n (GB)	Availa ble Memo ry (GB)	Shar ds (Mas ter Nod es)	Max. Connect ions (Default /Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performanc e (QPS)	Specification Code (spec_code in the API)
4	4	3	20,000/2 0,000	2,304/2,3 04	x86: 300,000 Arm: 240,000	x86: redis.proxy.xu1.l arge.4 Arm: redis.proxy.au1.l arge.4
8	8	3	30,000/3 0,000	2,304/2,3 04	x86: 300,000 Arm: 240,000	x86: redis.proxy.xu1.l arge.8 Arm: redis.proxy.au1.l arge.8
16	16	3	30,000/3 0,000	2,304/2,3 04	x86: 300,000 Arm: 240,000	x86: redis.proxy.xu1.l arge.16 Arm: redis.proxy.au1.l arge.16

Specification (GB)	Availa ble Memo ry (GB)	Shar ds (Mas ter Nod es)	Max. Connect ions (Default /Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performanc e (QPS)	Specification Code (spec_code in the API)
24	24	3	30,000/3 0,000	2,304/2,3 04	x86: 300,000 Arm: 240,000	x86: redis.proxy.xu1.l arge.24 Arm: redis.proxy.au1.l arge.24
32	32	3	30,000/3 0,000	2,304/2,3 04	x86: 300,000 Arm: 240,000	x86: redis.proxy.xu1.l arge.32 Arm: redis.proxy.au1.l arge.32
48	48	6	60,000/6 0,000	4,608/4,6 08	x86: 600,000 Arm: 480,000	x86: redis.proxy.xu1.l arge.48 Arm: redis.proxy.au1.l arge.48
64	64	8	80,000/8 0,000	6,144/6,1 44	x86: 800,000 Arm: 640,000	x86: redis.proxy.xu1.l arge.64 Arm: redis.proxy.au1.l arge.64
96	96	12	120,000/ 120,000	9,216/9,2 16	x86: 1,200,000 Arm: 960,000	x86: redis.proxy.xu1.l arge.96 Arm: redis.proxy.au1.l arge.96
128	128	16	160,000/ 160,000	10,000/1 0,000	x86: 1,600,000 Arm: 1,280,000	x86: redis.proxy.xu1.l arge.128 Arm: redis.proxy.au1.l arge.128

Specification (GB)	Availa ble Memo ry (GB)	Shar ds (Mas ter Nod es)	Max. Connect ions (Default /Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performanc e (QPS)	Specification Code (spec_code in the API)
192	192	24	200,000/ 240,000	10,000/1 0,000	x86: 2,400,000 Arm: 1,920,000	x86: redis.proxy.xu1.l arge.192 Arm: redis.proxy.au1.l arge.192
256	256	32	200,000/ 320,000	10,000/1 0,000	x86: 3,200,000 Arm: 2,560,000	x86: redis.proxy.xu1.l arge.256 Arm: redis.proxy.au1.l arge.256
384	384	48	200,000/ 480,000	10,000/1 0,000	x86: 4,800,000 Arm: 3,840,000	x86: redis.proxy.xu1.l arge.384 Arm: redis.proxy.au1.l arge.384
512	512	64	200,000/ 500,000	10,000/1 0,000	x86: 6,400,000 Arm: 5,120,000	x86: redis.proxy.xu1.l arge.512 Arm: redis.proxy.au1.l arge.512
768	768	96	200,000/ 500,000	10,000/1 0,000	x86: 9,600,000 Arm: 7,680,000	x86: redis.proxy.xu1.l arge.768 Arm: redis.proxy.au1.l arge.768
1024	1024	128	200,000/ 500,000	10,000/1 0,000	x86: 12,800,000 Arm: 10,240,000	x86: redis.proxy.xu1.l arge.1024 Arm: redis.proxy.au1.l arge.1024

Specification (GB)	Availa ble Memo ry (GB)	Shar ds (Mas ter Nod es)	Max. Connect ions (Default /Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Reference Performanc e (QPS)	Specification Code (spec_code in the API)
2048	2048	128	200,000/ 500,000	10,000/1 0,000	x86: 12,800,000 Arm: 10,240,000	x86: redis.proxy.xu1.l arge.2048 Arm: redis.proxy.au1.l arge.2048
4096	4096	128	200,000/ 500,000	10,000/1 0,000	x86: 12,800,000 Arm: 10,240,000	x86: redis.proxy.xu1.l arge.4096 Arm: redis.proxy.au1.l arge.4096

Redis Cluster Instances

Redis Cluster instances can have 1 to 5 replicas. For example, the instance specifications can be **Redis Cluster | 1 replica** or **Redis Cluster | 5 replicas**. By default, a Redis Cluster instance has two replicas. A Redis Cluster instance with only 1 replica indicates that the replica quantity has been decreased.

Given the same memory size, the differences between x86-based Redis Cluster instances, Arm-based Redis Cluster instances, and Redis Cluster instances with different replicas are as follows:

- The available memory, shard quantity (master node quantity), maximum number of connections, assured/maximum bandwidth, and QPS are the same.
- Instance flavor code (specification name): **Table 7-4** only lists the specification names of x86- and Arm-based instances with 2 replicas. The specification names reflect the number of replicas, for example, redis.cluster.xu1.large.**r2**.24 (Redis Cluster | x86 | 2 replicas | 24 GB) and redis.cluster.xu1.large.**r3**.24 (Redis Cluster | x86 | 3 replicas | 24 GB).
- IP addresses: Number of occupied IP addresses = Number of shards × Number of replicas. For example:
 - 24 GB | Redis Cluster | 3 replicas: Number of occupied IP addresses = 3 x 3 = 9
- Available memory per node = Instance available memory/Master node quantity
 - For example, a 24 GB x86-based instance has 24 GB available memory and 3 master nodes. The available memory per node is 24/3 = 8 GB.
- **Maximum connections limit per node** = Maximum connections limit/Master node quantity For example:

For example, a 24 GB x86-based instance has 3 master nodes and the maximum connections limit is 150,000. The maximum connections limit per node = 150,000/3 = 50,000.

□ NOTE

- The following table lists only the Redis Cluster instance specifications with default shards. If you customize shards, see the maximum number of connections, assured/ maximum bandwidth, and product specification code (flavor) in the Instance Specification table the Buy DCS Instance page of the DCS console.
- The maximum connections of a cluster is for the entire instance, and not for a single shard. Maximum connections of a single shard = Maximum connections of an instance/ Number of shards.
- The maximum bandwidth and assured bandwidth of a cluster is for the entire instance, and not for a single shard. The relationship between the instance bandwidth and the bandwidth of a single shard is as follows:
 - Instance bandwidth = Bandwidth of a single shard x Number of shards
 - For a cluster instance, if the memory of a single shard is 1 GB, the bandwidth of a single shard is 384 Mbit/s. If the memory of a single shard is greater than 1 GB, the bandwidth of a single shard is 768 Mbit/s.

Table 7-4 Specifications of cluster DCS Redis 4.0/5.0/6.0/7.0 instances

Total Memor y (GB)	Availabl e Memory (GB)	Shar ds (Mas ter Node s)	Max. Conne ctions (Defau lt/ Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Referenc e Performa nce (QPS)	Specification Code (spec_code in the API)
4	4	3	30,000 / 150,00 0	2304/23 04	x86: 300,000 Arm: 240,000	x86: redis.cluster.xu1.la rge.r2.4 Arm: redis.cluster.au1.l arge.r2.4
8	8	3	30,000 / 150,00 0	2304/23 04	x86: 300,000 Arm: 240,000	x86: redis.cluster.xu1.la rge.r2.8 Arm: redis.cluster.au1.l arge.r2.8
16	16	3	30,000 / 150,00 0	2304/23 04	x86: 300,000 Arm: 240,000	x86: redis.cluster.xu1.la rge.r2.16 Arm: redis.cluster.au1.l arge.r2.16

Total Memor y (GB)	Availabl e Memory (GB)	Shar ds (Mas ter Node s)	Max. Conne ctions (Defau lt/ Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Referenc e Performa nce (QPS)	Specification Code (spec_code in the API)
24	24	3	30,000 / 150,00 0	2304/23 04	x86: 300,000 Arm: 240,000	x86: redis.cluster.xu1.la rge.r2.24 Arm: redis.cluster.au1.l arge.r2.24
32	32	3	30,000 / 150,00 0	2304/23 04	x86: 300,000 Arm: 240,000	x86: redis.cluster.xu1.la rge.r2.32 Arm: redis.cluster.au1.l arge.r2.32
48	48	6	60,000 / 300,00 0	4608/46 08	x86: 600,000 Arm: 480,000	x86: redis.cluster.xu1.la rge.r2.48 Arm: redis.cluster.au1.l arge.r2.48
64	64	8	80,000 / 400,00 0	6144/61 44	x86: 800,000 Arm: 640,000	x86: redis.cluster.xu1.la rge.r2.64 Arm: redis.cluster.au1.l arge.r2.64
96	96	12	120,00 0 / 600,00 0	9216/92 16	x86: 1,200,000 Arm: 960,000	x86: redis.cluster.xu1.la rge.r2.96 Arm: redis.cluster.au1.l arge.r2.96
128	128	16	160,00 0 / 800,00 0	12,288/1 2,288	x86: 1,600,000 Arm: 1,280,000	x86: redis.cluster.xu1.la rge.r2.128 Arm: redis.cluster.au1.l arge.r2.128

Total Memor y (GB)	Availabl e Memory (GB)	Shar ds (Mas ter Node s)	Max. Conne ctions (Defau lt/ Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Referenc e Performa nce (QPS)	Specification Code (spec_code in the API)
192	192	24	240,00 0 / 1,200,0 00	18,432/1 8,432	x86: 2,400,000 Arm: 1,920,000	x86: redis.cluster.xu1.la rge.r2.192 Arm: redis.cluster.au1.l arge.r2.192
256	256	32	320,00 0 / 1,600,0 00	24,576/2 4,576	x86: 3,200,000 Arm: 2,560,000	x86: redis.cluster.xu1.la rge.r2.256 Arm: redis.cluster.au1.l arge.r2.256
384	384	48	480,00 0 / 2,400,0 00	36,864/3 6,864	x86: 4,800,000 Arm: 3,840,000	x86: redis.cluster.xu1.la rge.r2.384 Arm: redis.cluster.au1.l arge.r2.384
512	512	64	640,00 0 / 3,200,0 00	49,152/4 9,152	x86: 6,400,000 Arm: 5,120,000	x86: redis.cluster.xu1.la rge.r2.512 Arm: redis.cluster.au1.l arge.r2.512
768	768	96	960,00 0 / 4,800,0 00	73,728/7 3,728	x86: 9,600,000 Arm: 7,680,000	x86: redis.cluster.xu1.la rge.r2.768 Arm: redis.cluster.au1.l arge.r2.768
1024	1024	128	1,280,0 00 / 6,400,0 00	98,304/9 8,304	x86: 12,800,00 0 Arm: 10,240,00	x86: redis.cluster.xu1.la rge.r2.1024 Arm: redis.cluster.au1.l arge.r2.1024

Total Memor y (GB)	Availabl e Memory (GB)	Shar ds (Mas ter Node s)	Max. Conne ctions (Defau lt/ Limit) (Count)	Assured/ Maximu m Bandwid th (Mbit/s)	Referenc e Performa nce (QPS)	Specification Code (spec_code in the API)
2048	2048	128	1,280,0 00 / 6,400,0 00	98,304/9 8,304	x86: 12,800,00 0 Arm: 10,240,00 0	x86: redis.cluster.xu1.la rge.r2.2048 Arm: redis.cluster.au1.l arge.r2.2048

Read/Write Splitting Instances

- **Table 7-5** describes the specifications of read/write splitting x86-based instances.
- The maximum connections of a read/write splitting DCS Redis instance cannot be modified.
- Bandwidth limit per Redis Server (MB/s) = Total bandwidth limit (MB/s)/
 Number of replicas (including masters)
- Reference performance (QPS) = Reference performance per node (QPS) × Number of replicas (including masters)
- When using read/write splitting instances, note the following:
 - a. Read requests are sent to replicas. There is a delay when data is synchronized from the master to the replicas.
 - If your services are sensitive to the delay, do not use read/write splitting instances. Instead, you can use master/standby or cluster instances.
 - b. Read/write splitting is suitable when there are more read requests than write requests. If there are a lot of write requests, the master and replicas may be disconnected, or the data synchronization between them may fail after the disconnection. As a result, the read performance deteriorates.
 - If your services are write-heavy, use master/standby or cluster instances.
 - c. If a replica is faulty, it takes some time to synchronize all data from the master. During the synchronization, the replica does not provide services, and the read performance of the instance deteriorates.
 - To reduce the impact of the interruption, use an instance with less than 32 GB memory. The smaller the memory, the shorter the time for full data synchronization between the master and replicas, and the smaller the impact of the interruption.

Table 7-5 Specifications of read/write splitting DCS Redis 4.0/5.0/6.0 instances

Spe cific atio n	Availa ble Memo ry (GB)	Replic as (Inclu ding Maste rs)	Max. Conne ctions (Defa ult/ Limit)	Bandw idth Limit (MB/s)	Bandw idth Limit per Redis Server (MB/s	Refere nce Perfor mance (QPS)	Refere nce Perfor mance per Node (QPS)	Specifi cation Code (spec_ code in the API)
1	1	2	20,000	96	48	200,00 0	100,00 0	redis.h a.xu1.l arge.p 2.1
1	1	3	30,000	144	48	300,00 0	100,00 0	redis.h a.xu1.l arge.p 3.1
1	1	4	40,000	192	48	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.1
1	1	5	50,000	240	48	500,00 0	100,00	redis.h a.xu1.l arge.p 5.1
1	1	6	60,000	288	48	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.1
2	2	2	20,000	96	48	200,00	100,00 0	redis.h a.xu1.l arge.p 2.2
2	2	3	30,000	144	48	300,00	100,00 0	redis.h a.xu1.l arge.p 3.2
2	2	4	40,000	192	48	400,00 0	100,00	redis.h a.xu1.l arge.p 4.2
2	2	5	50,000	240	48	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.2

Spe cific atio n	Availa ble Memo ry (GB)	Replic as (Inclu ding Maste rs)	Max. Conne ctions (Defa ult/ Limit)	Bandw idth Limit (MB/s)	Bandw idth Limit per Redis Server (MB/s)	Refere nce Perfor mance (QPS)	Refere nce Perfor mance per Node (QPS)	Specifi cation Code (spec_ code in the API)
2	2	6	60,000	288	48	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.2
4	4	2	20,000	96	48	200,00	100,00	redis.h a.xu1.l arge.p 2.4
4	4	3	30,000	144	48	300,00 0	100,00 0	redis.h a.xu1.l arge.p 3.4
4	4	4	40,000	192	48	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.4
4	4	5	50,000	240	48	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.4
4	4	6	60,000	288	48	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.4
8	8	2	20,000	192	96	200,00	100,00 0	redis.h a.xu1.l arge.p 2.8
8	8	3	30,000	288	96	300,00 0	100,00 0	redis.h a.xu1.l arge.p 3.8
8	8	4	40,000	384	96	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.8

Spe cific atio n	Availa ble Memo ry (GB)	Replic as (Inclu ding Maste rs)	Max. Conne ctions (Defa ult/ Limit)	Bandw idth Limit (MB/s)	Bandw idth Limit per Redis Server (MB/s)	Refere nce Perfor mance (QPS)	Refere nce Perfor mance per Node (QPS)	Specifi cation Code (spec_ code in the API)
8	8	5	50,000	480	96	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.8
8	8	6	60,000	576	96	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.8
16	16	2	20,000	192	96	200,00	100,00	redis.h a.xu1.l arge.p 2.16
16	16	3	30,000	288	96	300,00 0	100,00 0	redis.h a.xu1.l arge.p 3.16
16	16	4	40,000	384	96	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.16
16	16	5	50,000	480	96	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.16
16	16	6	60,000	576	96	600,00 0	100,00	redis.h a.xu1.l arge.p 6.16
32	32	2	20,000	192	96	200,00	100,00 0	redis.h a.xu1.l arge.p 2.32
32	32	3	30,000	288	96	300,00 0	100,00 0	redis.h a.xu1.l arge.p 3.32

Spe cific atio n	Availa ble Memo ry (GB)	Replic as (Inclu ding Maste rs)	Max. Conne ctions (Defa ult/ Limit)	Bandw idth Limit (MB/s)	Bandw idth Limit per Redis Server (MB/s)	Refere nce Perfor mance (QPS)	Refere nce Perfor mance per Node (QPS)	Specifi cation Code (spec_ code in the API)
32	32	4	40,000	384	96	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.32
32	32	5	50,000	480	96	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.32
32	32	6	60,000	576	96	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.32
64	64	2	20,000	192	96	200,00	100,00 0	redis.h a.xu1.l arge.p 2.64
64	64	3	30,000	288	96	300,00	100,00 0	redis.h a.xu1.l arge.p 3.64
64	64	4	40,000	384	96	400,00 0	100,00 0	redis.h a.xu1.l arge.p 4.64
64	64	5	50,000	480	96	500,00 0	100,00 0	redis.h a.xu1.l arge.p 5.64
64	64	6	60,000	576	96	600,00 0	100,00 0	redis.h a.xu1.l arge.p 6.64

7.2 Redis 6.0 Enterprise Edition

This section describes DCS Redis 6.0 enterprise edition instance specifications, including the total memory, available memory, maximum number of connections allowed, maximum/assured bandwidth, and reference performance.

DCS Redis 6.0 enterprise edition is available with performance and storage types. The instance metrics are as follows:

- Used memory: You can check the memory usage of an instance by viewing the **Memory Usage** and **Used Memory** metrics.
- Maximum connections: The maximum number of connections allowed is the
 maximum number of clients that can be connected to an instance. To check
 the number of connections to an instance, view the Connected Clients
 metric. After an instance is created, you can change the maximum number of
 connections of the instance by modifying the maxclients parameter on the
 Instance Configuration > Parameters page on the console.
- QPS represents queries per second, which is the number of commands processed per second. For details about QPS testing, see the Performance White Paper.
- Bandwidth: You can view the Flow Control Times metric to check whether
 the bandwidth has exceeded the limit. You can also check the Bandwidth
 Usage metric. This metric is for reference only, because it may be higher than
 100%. For details, see Why Does Bandwidth Usage Exceed 100%?

□ NOTE

Enterprise edition DCS is no longer provided. Existing instances can be used.

Enterprise (Performance) Edition

Currently, DCS for Redis 6.0 enterprise (performance) edition supports master/standby instances based on x86 CPUs.

Table 7-6 Specifications of DCS Redis 6.0 enterprise (performance) edition instances

Total Memo ry (GB)	Availab le Memor y (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Reference Performan ce (QPS)	Specification Code (spec_code in the API)
8	8	10,000/50,0 00	1,536/1,53 6	400,000	redis.ha.xu1.large. enthp.8
16	16	10,000/50,0 00	1,536/1,53 6	400,000	redis.ha.xu1.large. enthp.16
32	32	10,000/50,0 00	1,536/1,53 6	400,000	redis.ha.xu1.large. enthp.32

Total Memo ry (GB)	Availab le Memor y (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Reference Performan ce (QPS)	Specification Code (spec_code in the API)
64	64	10,000/50,0 00	1,536/1,53 6	400,000	redis.ha.xu1.large. enthp.64

Enterprise (Storage) Edition

Currently, DCS for Redis 6.0 enterprise (storage) edition supports master/standby instances based on x86 CPUs.

Enterprise (storage) instances use memory and SSDs. They use memory to cache hot data and SSDs to store all data. "Available Memory" in the following table is the disk capacity.

Table 7-7 Specifications of DCS Redis 6.0 enterprise (storage) edition instances

Total Memo ry (GB)	Maximu m Storage (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Reference Performa nce (QPS)	Specification Code (spec_code in the API)
8	64	10,000/50,0 00	768/768	70,000	redis.ha.xu1.large. entst.8
16	128	10,000/50,0 00	768/768	70,000	redis.ha.xu1.large. entst.16
32	256	10,000/50,0 00	768/768	70,000	redis.ha.xu1.large. entst.32

7.3 Redis 3.0 Instance Specifications (Discontinued)

This section describes DCS Redis 3.0 instance specifications, including the total memory, available memory, maximum number of connections allowed, assured bandwidth, and reference performance.

The following metrics are related to the instance specifications:

- Used memory: You can check the memory usage of an instance by viewing the **Memory Usage** and **Used Memory** metrics.
- Maximum connections: The maximum number of connections allowed is the maximum number of clients that can be connected to an instance. To check the number of connections to an instance, view the Connected Clients metric.

• QPS represents queries per second, which is the number of commands processed per second.

■ NOTE

- DCS Redis 3.0 instances are available in single-node, master/standby, and Proxy Cluster types.
- DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 5.0 or later instead.

Single-Node Instances

For each single-node DCS Redis instance, the available memory is less than the total memory because some memory is reserved for system overheads, as shown in **Table 7-8**.

Table 7-8 Specifications of single-node DCS Redis 3.0 instances

Total Memor y (GB)	Availabl e Memory (GB)	Max. Connectio ns (Default/ Limit) (Count)	Assured Bandwidth (Mbit/s)	Reference Performa nce (QPS)	Specification Code (spec_code in the API)
2	1.5	5000/50,00 0	42	50,000	dcs.single_node
4	3.2	5000/50,00 0	64	100,000	dcs.single_node
8	6.8	5000/50,00 0	64	100,000	dcs.single_node
16	13.6	5000/50,00 0	85	100,000	dcs.single_node
32	27.2	5000/50,00 0	85	100,000	dcs.single_node
64	58.2	5000/60,00 0	128	100,000	dcs.single_node

Master/Standby Instances

For each master/standby DCS Redis instance, the available memory is less than that of a single-node DCS Redis instance because some memory is reserved for data persistence, as shown in **Table 7-9**. The available memory of a master/standby instance can be adjusted to support background tasks such as data persistence and master/standby synchronization.

Assured **Specification** Total Availabl Max. Reference **Bandwidt** Code (spec_code Memory Connectio Performan in the API) Memory ns ce (GB) (Default/ (Mbit/s) (GB) (QPS) Limit) (Count) 2 1.5 5000/50,00 50,000 42 dcs.master_stand by 4 3.2 5000/50,00 64 100,000 dcs.master_stand by 5000/50,00 8 6.4 64 100,000 dcs.master stand by 16 12.8 5000/50,00 85 100,000 dcs.master_stand by 32 25.6 5000/50,00 85 100,000 dcs.master_stand by 5000/60,00 64 51.2 128 100,000 dcs.master_stand by

Table 7-9 Specifications of master/standby DCS Redis 3.0 instances

Proxy Cluster Instances

In addition to larger memory, Proxy Cluster instances feature more connections allowed, higher bandwidth allowed, and more QPS than single-node and master/standby instances.

Table 7-10 Specifications of Proxy Cluster DCS Redis 3.0 instances

Total Memory (GB)	Availabl e Memory (GB)	Max. Connectio ns (Default/ Limit) (Count)	Assured Bandwidt h (Mbit/s)	Reference Performan ce (QPS)	Specification Code (spec_code in the API)
64	64	90,000/90, 000	600	500,000	dcs.cluster
128	128	180,000/18 0,000	600	500,000	dcs.cluster
256	256	240,000/24 0,000	600	500,000	dcs.cluster
512	512	480,000/48 0,000	600	500,000	dcs.cluster

Total Memory (GB)	Availabl e Memory (GB)	Max. Connectio ns (Default/ Limit) (Count)	Assured Bandwidt h (Mbit/s)	Reference Performan ce (QPS)	Specification Code (spec_code in the API)
1024	1024	960,000/96 0,000	600	500,000	dcs.cluster

■ NOTE

- Pay-per-use Proxy Cluster DCS Redis instances are available in 64 GB, 128 GB, and 256 GB.
- Yearly/Monthly Proxy Cluster DCS Redis instances are available in 64 GB, 128 GB, 256 GB, 512 GB, and 1024 GB.

Currently, only the CN-Hong Kong region supports yearly/monthly billing. If you need to use this billing mode in other regions, submit a service ticket on the console to request the technical personnel to enable the function for you in the background.

7.4 Memcached Instance Specifications (Discontinued)

□ NOTE

DCS for Memcached is no longer provided. You can use DCS Redis instances instead.

This section describes DCS Memcached instance specifications, including the total memory, available memory, maximum number of connections allowed, maximum/ assured bandwidth, and reference performance.

The maximum number of connections allowed is the maximum number of clients connected to an instance. To check the number of connections to an instance, view the **Connected Clients** metric.

QPS represents queries per second, which is the number of commands processed per second.

DCS Memcached instances are available in single-node and master/standby types.

Single-Node Instances

For each single-node DCS Memcached instance, the available memory is less than the total memory because some memory is reserved for system overheads, as shown in **Table 7-11**.

Total Memory (GB)	Available Memory (GB)	Max. Connectio ns (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Referen ce Perfor mance (QPS)	Specification Code (spec_code in the API)
2	1.5	5000/50,00 0	42/128	50,000	dcs.memcached.si ngle_node
4	3.2	5000/50,00 0	64/192	100,000	dcs.memcached.si ngle_node
8	6.8	5000/50,00 0	64/192	100,000	dcs.memcached.si ngle_node
16	13.6	5000/50,00 0	85/256	100,000	dcs.memcached.si ngle_node
32	27.2	5000/50,00 0	85/256	100,000	dcs.memcached.si ngle_node
64	58.2	5000/50,00 0	128/384	100,000	dcs.memcached.si ngle_node

Table 7-11 Specifications of single-node DCS Memcached instances

Master/Standby Instances

For each master/standby DCS Memcached instance, the available memory is less than the total memory because some memory is reserved for data persistence, as shown in **Table 7-12**. The available memory of a master/standby instance can be adjusted to support background tasks such as data persistence and master/standby synchronization.

Table 7-12 Specifications of master/standby DCS Memcached instances

Total Memory (GB)	Available Memory (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Referen ce Perfor mance (QPS)	Specification Code (spec_code in the API)
2	1.5	5000/50,00 0	42/128	50,000	dcs.memcached. master_standby
4	3.2	5000/50,00 0	64/192	100,000	dcs.memcached. master_standby
8	6.8	5000/50,00 0	64/192	100,000	dcs.memcached. master_standby

Total Memory (GB)	Available Memory (GB)	Max. Connection s (Default/ Limit) (Count)	Assured/ Maximum Bandwidth (Mbit/s)	Referen ce Perfor mance (QPS)	Specification Code (spec_code in the API)
16	13.6	5000/50,00 0	85/256	100,000	dcs.memcached. master_standby
32	27.2	5000/50,00 0	85/256	100,000	dcs.memcached. master_standby
64	58.2	5000/50,00 0	128/384	100,000	dcs.memcached. master_standby

8 Command Compatibility

8.1 Commands Supported and Disabled by DCS for Redis 4.0

DCS for Redis 4.0 is developed based on Redis 4.0.14 and is compatible with open-source protocols and commands. This section describes DCS for Redis 4.0's compatibility with Redis commands, including supported and disabled commands.

DCS Redis instances support most Redis commands. Any client compatible with the Redis protocol can access DCS.

- For security purposes, some Redis commands are disabled in DCS, as listed in Commands Disabled by DCS for Redis 4.0.
- Some Redis commands are supported by cluster DCS instances for multi-key operations in the same slot. For details, see **Command Restrictions**.
- Some Redis commands (such as KEYS, FLUSHDB, and FLUSHALL) have usage restrictions, which are described in Other Command Usage Restrictions.
- Some high-risk commands can be renamed. For details, see Commands That Can Be Renamed.

Commands Supported by DCS for Redis 4.0

- Table 8-1 and Table 8-2 list commands supported by single-node, master/ standby, and Redis Cluster DCS Redis 4.0 instances.
- Table 8-3 and Table 8-4 list the Redis commands supported by Proxy Cluster DCS Redis 4.0 instances.
- Table 8-5 and Table 8-6 list the Redis commands supported by read/write splitting DCS Redis 4.0 instances.

For details about the command syntax, visit the **Redis official website**. For example, to view details about the **SCAN** command, enter **SCAN** in the search box on **this page**.

■ NOTE

- Commands available since later Redis versions are not supported by earlier-version instances. Run a command on redis-cli to check whether it is supported by DCS for Redis. If the message "(error) ERR unknown command" is returned, the command is not supported.
- For DCS Redis 4.0 instances in the Redis Cluster mode, ensure that all commands in a pipeline are executed on the same shard.

Table 8-1 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 4.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEN D	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOU NT	HEXIST S	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOP LRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETAL L	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRB Y	LINSER T	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRB YFLOAT	LLEN	SINTERS TORE	ZRANGEBYS CORE	CLIENT KILL
PTTL	GET	HKEYS	LPOP	SISMEM BER	ZRANK	CLIENT LIST
RANDO MKEY	GETRA NGE	HMGET	LPUSH X	SMEMBE RS	ZREMRANGE BYRANK	CLIENT GETNAME
RENAME	GETSET	HMSET	LRANG E	SMOVE	ZREMRANGE BYCORE	CLIENT SETNAME
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRANGE	CONFIG GET
RESTOR E	INCRBY	HSETN X	LSET	SRAND MEMBE R	ZREVRANGE BYSCORE	MONITOR
SORT	INCRBY FLOAT	HVALS	LTRIM	SREM	ZREVRANK	SLOWLOG
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	ROLE
TYPE	MSET	HSTRLE N	RPOPL PU	SUNION STORE	ZUNIONSTO RE	SWAPDB

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
SCAN	MSETN X	HLEN	RPOPL PUSH	SSCAN	ZINTERSTOR E	MEMORY
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	CONFIG
PEXPIRE	SET	-	RPUSH X	-	ZRANGEBYL EX	COMMAN D
PEXPIRE AT	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
KEYS	SETEX	-	-	-	ZREMRANGE BYSCORE	-
-	SETNX	-	-	-	ZREM	-
-	SETRAN GE	-	-	-	-	-
-	STRLEN	-	-	-	-	-
-	BITFIEL D	-	-	-	-	-

Table 8-2 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 4.0 instances (2)

HyperLogL og	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo
PFADD	PSUBSCRI BE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBS CRIBE	UNWATC H	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIB E	WATCH	SELECT (not supporte d by Redis Cluster instances)	SCRIPT KILL	GEORADIUS
-	UNSUBSC RIBE	-	-	SCRIPT LOAD	GEORADIUSBY MEMBER

Table 8-3 Commands supported by Proxy Cluster DCS Redis 4.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	MEMORY
RENAME	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	COMMA ND
RENAME NX	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	COMMA ND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRA NGE	COMMA ND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND INFO
TTL	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETST AT

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
SCAN	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	CONFIG REWRITE
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	-	ZSCAN	-
PEXPIREA T	SET	-	LPUSH	-	ZRANGE BYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOU NT	-
KEYS	SETEX	-	-	-	ZREMRA NGEBYSC ORE	-
TOUCH	SETNX	-	-	-	ZREM	-
UNLINK	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-
RANDO MKEY	STRLEN	-	-	-	ZREVRA NGEBYLE X	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Table 8-4 Commands supported by Proxy Cluster DCS Redis 4.0 instances (2)

HyperL ogLog	Pub/Sub	Transact ions	Connect ion	Scripting	Geo	Cluster
PFADD	PSUBSC RIBE	DISCAR D	AUTH	EVAL	GEOADD	CLUSTER INFO
PFCOU NT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	CLUSTER NODES
PFMER GE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	CLUSTER SLOTS
-	PUNSUB SCRIBE	UNWAT CH	QUIT	SCRIPT FLUSH	GEODIST	CLUSTER ADDSLOTS
-	SUBSCRI BE	WATCH	SELECT	SCRIPT KILL	GEORADIU S	ASKING

HyperL ogLog	Pub/Sub	Transact ions	Connect ion	Scripting	Geo	Cluster
-	UNSUBS CRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIU SBYMEMBE R	READONL Y
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEARC H	READWRI TE
-	-	-	CLIENT GETNAM E	-	GEOSEARC HSTORE	-
-	-	-	CLIENT SETNAM E	-	-	-

□ NOTE

Cluster commands in the preceding table are supported only by Proxy Cluster instances created on or after September 1, 2022.

Table 8-5 Commands supported by read/write splitting DCS Redis 4.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	MONITO R

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	SLOWLO G
RANDO MKEY	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	SWAPDB
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRA NGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND
SORT	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	COMMA ND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMA ND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	COMMA ND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG GET
OBJECT	PSETEX	-	RPUSHX	-	ZSCAN	CONFIG RESETST AT
PEXPIRE	SET	-	LPUSH	-	ZRANGE BYLEX	CONFIG REWRITE
PEXPIREA T	SETBIT	-	-	-	ZLEXCOU NT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRA NGEBYSC ORE	-
KEYS	SETNX	_	-	-	ZREM	-
TOUCH	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
UNLINK	STRLEN	-	-	-	ZREVRA NGEBYLE X	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Table 8-6 Commands supported by read/write splitting DCS Redis 4.0 instances (2)

HyperLogL og	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo
PFADD	PSUBSCRI BE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBS CRIBE	UNWATC H	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIB E	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSC RIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBY MEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEARCH
-	-	-	CLIENT GETNAM E	-	GEOSEARCHST ORE
-	-	-	CLIENT SETNAM E	-	-

Commands Disabled by DCS for Redis 4.0

The following lists commands disabled by DCS for Redis 4.0.

Table 8-7 Redis commands disabled in single-node and master/standby Redis 4.0 instances

Generic (Key)	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

Table 8-8 Redis commands disabled in Proxy Cluster DCS Redis 4.0 instances

Generic (Key)	Server	Sorted Set
MIGRATE	BGREWRITEAOF	ВZРОРМАХ
MOVE	BGSAVE	BZPOPMIN
WAIT	CLIENT commands	ZPOPMAX
-	DEBUG OBJECT	ZPOPMIN
-	DEBUG SEGFAULT	-
-	LASTSAVE	-
-	PSYNC	-
-	SAVE	-
-	SHUTDOWN	-
-	SLAVEOF	-
-	LATENCY commands	-
-	MODULE commands	-
-	LOLWUT	-
-	SWAPDB	-
-	REPLICAOF	-
-	SYNC	-

Table 8-9 Redis commands disabled in Redis Cluster DCS Redis 4.0 instances

Generic (Key)	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG commands	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT-FAILURE- REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Table 8-10 Redis commands disabled in read/write splitting DCS Redis 4.0 instances

Generic	Server	Sorted Set
MIGRATE	BGREWRITEAOF	BZPOPMAX
WAIT	BGSAVE	BZPOPMIN
-	DEBUG OBJECT	ZPOPMAX
-	DEBUG SEGFAULT	ZPOPMIN
-	LASTSAVE	-
-	LOLWUT	-
-	MODULE LIST/LOAD/ UNLOAD	-
-	PSYNC	-
-	REPLICAOF	-
-	SAVE	-

Generic	Server	Sorted Set
-	SHUTDOWN [NOSAVE SAVE]	-
-	SLAVEOF	-
-	SWAPDB	-
-	SYNC	-

Commands That Can Be Renamed

Table 8-11 Commands that can be renamed

Command	command, keys, flushdb, flushall, hgetall, scan, hscan, sscan, and zscan For Proxy Cluster instances, the dbsize and dbstats commands can also be renamed.
Method	See Renaming Commands.

8.2 Commands Supported and Disabled by DCS for Redis 5.0

DCS for Redis 5.0 is developed based on Redis 5.0.9 and is compatible with open-source protocols and commands. This section describes DCS for Redis 5.0's compatibility with Redis commands, including supported and disabled commands.

DCS Redis instances support most Redis commands. Any client compatible with the Redis protocol can access DCS.

- For security purposes, some Redis commands are disabled in DCS, as listed in Commands Disabled by DCS for Redis 5.0.
- Some Redis commands are supported by cluster DCS instances for multi-key operations in the same slot. For details, see Command Restrictions.
- Some Redis commands (such as KEYS, FLUSHDB, and FLUSHALL) have usage restrictions, which are described in Other Command Usage Restrictions.
- Some high-risk commands can be renamed. For details, see Commands That Can Be Renamed.

Commands Supported by DCS for Redis 5.0

- Table 8-12 and Table 8-13 list commands supported by single-node, master/ standby, and Redis Cluster DCS Redis 5.0 instances.
- Table 8-14 and Table 8-15 list commands supported by Proxy Cluster DCS for Redis 5.0 instances.

• Table 8-16 and Table 8-17 list the Redis commands supported by read/write splitting DCS Redis 5.0 instances.

For details about the command syntax, visit the **Redis official website**. For example, to view details about the **SCAN** command, enter **SCAN** in the search box on **this page**.

◯ NOTE

- Commands available since later Redis versions are not supported by earlier-version instances. Run a command on redis-cli to check whether it is supported by DCS for Redis. If the message "(error) ERR unknown command" is returned, the command is not supported.
- For DCS Redis 5.0 instances in the Redis Cluster mode, ensure that all commands in a pipeline are executed on the same shard.

Table 8-12 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 5.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEN D	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOU NT	HEXIST S	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOP LRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETAL L	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRB Y	LINSER T	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRB YFLOAT	LLEN	SINTERS TORE	ZRANGEBYS CORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEM BER	ZRANK	CLIENT KILL
RANDO MKEY	GETRA NGE	HMGET	LPUSH X	SMEMBE RS	ZREMRANGE BYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANG E	SMOVE	ZREMRANGE BYCORE	CLIENT GETNAME
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTOR E	INCRBY	HSETN X	LSET	SRAND MEMBE R	ZREVRANGE BYSCORE	CONFIG GET

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
SORT	INCRBY FLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLE N	RPOPL PU	SUNION STORE	ZUNIONSTO RE	ROLE
SCAN	MSETN X	HLEN	RPOPL PUSH	SSCAN	ZINTERSTOR E	SWAPDB
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	MEMORY
PEXPIRE AT	SET	-	RPUSH X	-	ZRANGEBYL EX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	COMMAN D
KEYS	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRAN GE	-	-	-	ZREMRANGE BYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIEL D	-	-	-	-	-

Table 8-13 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 5.0 instances (2)

HyperLo gLog	Pub/Su b	Transac tions	Connec tion	Scriptin g	Geo	Stream
PFADD	PSUBSC RIBE	DISCAR D	AUTH	EVAL	GEOADD	XACK
PFCOUN T	PUBLIS H	EXEC	ECHO	EVALSH A	GEOHASH	XADD
PFMERG E	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSU BSCRIBE	UNWAT CH	QUIT	SCRIPT FLUSH	GEODIST	XDEL

HyperLo gLog	Pub/Su b	Transac tions	Connec tion	Scriptin g	Geo	Stream
-	SUBSCR IBE	WATCH	SELECT (not support ed by Redis Cluster instanc es)	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUB SCRIBE	-	-	SCRIPT LOAD	GEORADIUS BYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGR OUP
-	-	-	-	-	-	XREVRANG E
-	-	-	-	-	-	XTRIM

Table 8-14 Commands supported by Proxy Cluster DCS Redis 5.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	MEMORY
RENAME	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	COMMA ND
RENAME NX	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	COMMA ND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRA NGE	COMMA ND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND INFO
TTL	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETST AT
SCAN	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	CONFIG REWRITE
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	-	ZSCAN	-
PEXPIREA T	SET	-	LPUSH	-	ZRANGE BYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOU NT	-
KEYS	SETEX	-	-	-	ZREMRA NGEBYSC ORE	-
UNLINK	SETNX	-	-	-	ZREM	-
TOUCH	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
RANDO MKEY	STRLEN	-	-	-	ZPOPMA X	-
-	BITFIELD	-	-	-	ZPOPMI N	-
-	GETBIT	-	-	-	BZPOPM AX	-
-	-	-	-	-	BZPOPMI N	-
-	-	-	-	-	ZREVRA NGEBYLE X	-

Table 8-15 Commands supported by Proxy Cluster DCS Redis 5.0 instances (2)

Hyper LogLo g	Pub/Su b	Transa ctions	Connecti on	Scriptin g	Geo	Cluster
PFAD D	PSUBSC RIBE	DISCAR D	AUTH	EVAL	GEOADD	CLUSTER INFO
PFCO UNT	PUBLISH	EXEC	ECHO	EVALSH A	GEOHASH	CLUSTER NODES
PFME RGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	CLUSTER SLOTS
-	PUNSUB SCRIBE	UNWA TCH	QUIT	SCRIPT FLUSH	GEODIST	CLUSTER ADDSLOTS
-	SUBSCRI BE	WATC H	SELECT	SCRIPT KILL	GEORADIUS	ASKING
-	UNSUBS CRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSB YMEMBER	READONLY
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEARCH	READWRIT E
-	-	-	CLIENT GETNAM E	-	GEOSEARCH STORE	-

Hyper LogLo g		Transa ctions	Connecti on	Scriptin g	Geo	Cluster
-	-	-	CLIENT SETNAME	-	-	-

□ NOTE

Cluster commands in the preceding table are supported only by Proxy Cluster instances created on or after September 1, 2022.

Table 8-16 Commands supported by read/write splitting DCS Redis 5.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	MONITO R
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	SLOWLO G
RANDO MKEY	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	SWAPDB
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRA NGE	MEMORY

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
RESTORE	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND
SORT	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	COMMA ND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMA ND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	COMMA ND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG GET
OBJECT	PSETEX	-	RPUSHX	-	ZSCAN	CONFIG RESETST AT
PEXPIRE	SET	-	LPUSH	-	ZRANGE BYLEX	CONFIG REWRITE
PEXPIREA T	SETBIT	-	-	-	ZLEXCOU NT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRA NGEBYSC ORE	-
KEYS	SETNX	-	-	-	ZREM	-
UNLINK	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-
TOUCH	STRLEN	-	-	-	BZPOPM AX	-
-	BITFIELD	-	-	-	BZPOPMI N	-
-	GETBIT	-	-	-	ZPOPMA X	-
-	-	-	-	-	ZPOPMI N	-
-	-	-	-	-	ZREVRA NGEBYLE X	-

Table 8-17 Commands supported by read/write splitting DCS Redis 5.0 instances (2)

HyperLo gLog	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo	Stream
PFADD	PSUBSCR IBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUN T	PUBLISH	EXEC	ECHO	EVALSHA	GEOHAS H	XADD
PFMERG E	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUB SCRIBE	UNWATC H	QUIT	SCRIPT FLUSH	GEODIST	XDEL
-	SUBSCRI BE	WATCH	SELECT	SCRIPT KILL	GEORADI US	XGROUP
-	UNSUBS CRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADI USBYME MBER	XINFO
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEAR CH	XLEN
-	-	-	CLIENT GETNAM E	-	GEOSEAR CHSTOR E	XPENDIN G
-	-	-	CLIENT SETNAM E	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADG ROUP
-	-	-	-	-	-	XREVRA NGE
-	-	-	-	-	-	XTRIM

Commands Disabled by DCS for Redis 5.0

The following lists commands disabled by DCS for Redis 5.0.

Table 8-18 Redis commands disabled in single-node and master/standby Redis 5.0 instances

Generic (Key)	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

Table 8-19 Redis commands disabled in Proxy Cluster DCS Redis 5.0 instances

Generic (Key)	Server
MIGRATE	BGREWRITEAOF
MOVE	BGSAVE
WAIT	CLIENT commands
-	DEBUG OBJECT
-	DEBUG SEGFAULT
-	LASTSAVE
-	PSYNC
-	SAVE
-	SHUTDOWN
-	SLAVEOF
-	LATENCY commands
-	MODULE commands
-	LOLWUT
-	SWAPDB
-	REPLICAOF
-	SYNC

Table 8-20 Redis commands disabled in Redis Cluster DCS Redis 5.0 instances

Generic (Key)	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG commands	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT-FAILURE- REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Table 8-21 Redis commands disabled in read/write splitting DCS Redis 5.0 instances

Generic	Server
MIGRATE	BGREWRITEAOF
WAIT	BGSAVE
-	DEBUG OBJECT
-	DEBUG SEGFAULT
-	LASTSAVE
-	LOLWUT
-	MODULE LIST/LOAD/UNLOAD
-	PSYNC
-	REPLICAOF
-	SAVE
-	SHUTDOWN [NOSAVE SAVE]
-	SLAVEOF

Generic	Server
-	SWAPDB
-	SYNC

Commands That Can Be Renamed

Table 8-22 Commands that can be renamed

Command	command, keys, flushdb, flushall, hgetall, scan, hscan, sscan, and zscan For Proxy Cluster instances, the dbsize and dbstats commands can also be renamed.
Method	See Renaming Commands.

8.3 Commands Supported and Disabled by DCS for Redis 6.0

Huawei Cloud DCS for Redis 6.0 is fully compatible with open-source Redis 6.

This section describes DCS for Redis 6.0's compatibility with KeyDB commands, including supported and disabled commands.

DCS Redis instances support most Redis commands. Any client compatible with the Redis protocol can access DCS.

- For security purposes, some Redis commands are disabled in DCS, as listed in Commands Disabled by DCS for Redis 6.0.
- Some Redis commands are supported by cluster DCS instances for multi-key operations in the same slot. For details, see Command Restrictions.
- Some Redis commands (such as KEYS, FLUSHDB, and FLUSHALL) have usage restrictions, which are described in Other Command Usage Restrictions.
- Some high-risk commands can be renamed. For details, see Commands That Can Be Renamed.

Commands Supported by DCS for Redis 6.0 Basic Edition

- Table 8-23 and Table 8-24 list commands supported by single-node, master/ standby, and Redis Cluster DCS Redis 6.0 instances.
- Table 8-25 and Table 8-26 list commands supported by Proxy Cluster DCS for Redis 6.0 instances.
- Table 8-27 and Table 8-28 list the Redis commands supported by read/write splitting DCS Redis 6.0 instances.

For details about the command syntax, visit the **Redis official website**. For example, to view details about the **SCAN** command, enter **SCAN** in the search box on **this page**.

Table 8-23 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEN D	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOU NT	HEXIST S	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOP LRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETAL L	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRB Y	LINSER T	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRB YFLOAT	LLEN	SINTERS TORE	ZRANGEBYS CORE	CONFIG GET
PTTL	GET	HKEYS	LPOP	SISMEM BER	ZRANK	MONITOR
RANDO MKEY	GETRA NGE	HMGET	LPUSH X	SMEMBE RS	ZREMRANGE BYRANK	SLOWLOG
RENAME	GETSET	HMSET	LRANG E	SMOVE	ZREMRANGE BYCORE	ROLE
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRANGE	SWAPDB
RESTOR E	INCRBY	HSETN X	LSET	SRAND MEMBE R	ZREVRANGE BYSCORE	MEMORY
SORT	INCRBY FLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	ACL
TYPE	MSET	HSTRLE N	RPOPL PU	SUNION STORE	ZUNIONSTO RE	COMMAN D
SCAN	MSETN X	HLEN	RPOPL PUSH	SSCAN	ZINTERSTOR E	-
OBJECT	PSETEX	-	RPUSH	SMISME MBER	ZSCAN	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
PEXPIRE AT	SET	-	RPUSH X	-	ZRANGEBYL EX	-
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
KEYS	SETEX	-	BLMOV E	-	ZPOPMIN	-
СОРҮ	SETNX	-	LMOVE	-	ZPOPMAX	-
-	SETRAN GE	-	LPOS	-	ZREMRANGE BYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIEL D	-	-	-	ZDIFF	-
-	BITFIEL D_RO	-	-	-	ZDIFFSTORE	-
-	GETDEL	-	-	-	ZINTER	-
-	GETEX	-	-	-	ZMSCORE	-
-	-	-	-	-	ZRANDMEM BER	-
-	-	-	-	-	ZRANGESTO RE	-
-	-	-	-	-	ZUNION	-

Table 8-24 Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances (2)

HyperLo glog	Pub/Su b	Transac tions	Connec tion	Scriptin g	Geo	Stream
PFADD	PSUBSC RIBE	DISCAR D	AUTH	EVAL	GEOADD	XACK
PFCOUN T	PUBLIS H	EXEC	ECHO	EVALSH A	GEOHASH	XADD
PFMERG E	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSU BSCRIBE	UNWAT CH	QUIT	SCRIPT FLUSH	GEODIST	XDEL

HyperLo glog	Pub/Su b	Transac tions	Connec tion	Scriptin g	Geo	Stream
-	SUBSCR IBE	WATCH	SELECT (not support ed by Redis Cluster instanc es)	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUB SCRIBE	-	CLIENT CACHI NG	SCRIPT LOAD	GEORADIUS BYMEMBER	XINFO
-	-	-	CLIENT GETRE DIR	-	-	XLEN
-	-	-	CLIENT INFO	-	-	XPENDING
-	-	-	CLIENT TRACKI NG	-	-	XRANGE
-	-	-	CLIENT TRACKI NGINF O	-	-	XREAD
-	-	-	CLIENT UNPAU SE	-	-	XREADGR OUP
-	-	-	CLIENT KILL	-	-	XREVRANG E
-	-	-	CLIENT LIST	-	-	XTRIM
-	-	-	CLIENT GETNA ME	-	-	XAUTOCLA IM
-	-	-	CLIENT SETNA ME	-	-	XGROUP CREATECO NSUMER
-	-	-	HELLO	-	-	-
-	-	-	RESET	-	-	-

Table 8-25 Commands supported by Proxy Cluster DCS Redis 6.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	MEMORY
RENAME	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	COMMA ND
RENAME NX	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	COMMA ND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRA NGE	COMMA ND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND INFO
TTL	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETST AT
SCAN	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	CONFIG REWRITE

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	SMISME MBER	ZSCAN	-
PEXPIREA T	SET	HRANDFI ELD	LPUSH	-	ZRANGE BYLEX	-
EXPIREAT	SETBIT	-	BLMOVE	-	ZLEXCOU NT	-
KEYS	SETEX	-	LMOVE	-	ZREMRA NGEBYSC ORE	-
UNLINK	SETNX	-	LPOS	-	ZREM	-
TOUCH	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-
RANDO MKEY	STRLEN	-	-	-	ZPOPMA X	-
COPY	BITFIELD	-	-	-	ZPOPMI N	-
-	GETBIT	-	-	-	BZPOPM AX	-
-	BITFIELD _RO	-	-	-	BZPOPMI N	-
-	GETDEL	-	-	-	ZREVRA NGEBYLE X	-
-	GETEX	-	-	-	ZDIFF	-
-	-	-	-	-	ZDIFFST ORE	-
-	-	-	-	-	ZINTER	-
-	-	-	-	-	ZMSCOR E	-
-	-	-	-	-	ZRANDM EMBER	-
-	-	-	-	-	ZRANGE STORE	-
-	-	-	-	-	ZUNION	-

Table 8-26 Commands supported by Proxy Cluster DCS Redis 6.0 instances (2)

Hyper Loglo g	Pub/Su b	Transa ctions	Connecti on	Scriptin g	Geo	Cluster
PFAD D	PSUBSC RIBE	DISCAR D	AUTH	EVAL	GEOADD	CLUSTER INFO
PFCO UNT	PUBLISH	EXEC	ECHO	EVALSH A	GEOHASH	CLUSTER NODES
PFME RGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	CLUSTER SLOTS
-	PUNSUB SCRIBE	UNWA TCH	QUIT	SCRIPT FLUSH	GEODIST	CLUSTER ADDSLOTS
-	SUBSCRI BE	WATC H	SELECT	SCRIPT KILL	GEORADIUS	ASKING
-	UNSUBS CRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSB YMEMBER	READONLY
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEARCH	READWRIT E
-	-	-	CLIENT GETNAM E	-	GEOSEARCH STORE	-
-	-	-	CLIENT SETNAME	-	-	-
-	-	-	HELLO	-	-	-

Table 8-27 Commands supported by read/write splitting DCS Redis 6.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHAL L (FLUSHA LL SYNC not supporte d.)
DUMP	BITCOUN T	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBY FLOAT	LLEN	SINTERS TORE	ZRANGE BYSCORE	MONITO R
PTTL	GET	HKEYS	LPOP	SISMEMB ER	ZRANK	SLOWLO G
RANDO MKEY	GETRAN GE	HMGET	LPUSHX	SMEMBE RS	ZREMRA NGEBYR ANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRA NGEBYC ORE	SWAPDB
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRA NGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDM EMBER	ZREVRA NGEBYSC ORE	COMMA ND
SORT	INCRBYF LOAT	HVALS	LTRIM	SREM	ZREVRA NK	COMMA ND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMA ND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPU SH	SUNION STORE	ZUNION STORE	COMMA ND INFO

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERS TORE	CONFIG GET
OBJECT	PSETEX	HRANDFI ELD	RPUSHX	SMISME MBER	ZSCAN	CONFIG RESETST AT
PEXPIRE	SET	-	LPUSH	-	ZRANGE BYLEX	CONFIG REWRITE
PEXPIRE AT	SETBIT	-	BLMOVE	-	ZLEXCOU NT	CONFIG SET
EXPIREAT	SETEX	-	LMOVE	-	ZREMRA NGEBYSC ORE	-
KEYS	SETNX	-	LPOS	-	ZREM	-
UNLINK	SETRAN GE	-	-	-	ZREMRA NGEBYLE X	-
TOUCH	STRLEN	-	-	-	BZPOPM AX	-
COPY	BITFIELD	-	-	-	BZPOPMI N	-
-	GETBIT	-	-	-	ZPOPMA X	-
-	BITFIELD _RO	-	-	-	ZPOPMI N	-
-	GETDEL	-	-	-	ZREVRA NGEBYLE X	-
-	GETEX	-	-	-	ZDIFF	-
-	-	-	-	-	ZDIFFST ORE	-
-	-	-	-	-	ZINTER	-
-	-	-	-	-	ZMSCOR E	-
-	-	-	-	-	ZRANDM EMBER	-
-	-	-	-	-	ZRANGE STORE	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
-	-	-	-	-	ZUNION	-

Table 8-28 Commands supported by read/write splitting DCS Redis 6.0 instances (2)

HyperLo glog	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo	Stream
PFADD	PSUBSCR IBE	DISCARD	AUTH	EVAL	GEOADD	XAUTOC LAIM
PFCOUN T	PUBLISH	EXEC	ECHO	EVALSHA	GEOHAS H	XGROUP CREATEC ONSUME R
PFMERG E	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XACK
-	PUNSUB SCRIBE	UNWATC H	QUIT	SCRIPT FLUSH	GEODIST	XADD
-	SUBSCRI BE	WATCH	SELECT	SCRIPT KILL	GEORADI US	XCLAIM
-	UNSUBS CRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADI USBYME MBER	XDEL
-	-	-	CLIENT LIST	SCRIPT DEBUG YES SYNC NO	GEOSEAR CH	XGROUP
-	-	-	CLIENT GETNAM E	-	GEOSEAR CHSTOR E	XINFO
-	-	-	CLIENT SETNAM E	-	-	XLEN
-	-	-	HELLO	-	-	XPENDIN G
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADG ROUP

HyperLo glog	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo	Stream
-	-	-	-	-	-	XREVRA NGE
-	-	-	-	-	-	XTRIM

Commands Supported by DCS for Redis 6.0 Enterprise Edition

The following lists commands supported by DCS for Redis 6.0 enterprise edition.

Table 8-29 Commands supported by DCS for Redis 6.0 enterprise edition (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
COPY	APPEN D	HDEL	BLMOV E	SADD	BZPOPMAX	FLUSHALL
DEL	BITCOU NT	HEXIST S	LINDEX	SCARD	BZPOPMIN	FLUSHDB
DUMP	ВІТОР	HGET	LINSER T	SDIFF	ZADD	DBSIZE
EXISTS	BITPOS	HGETAL L	LLEN	SDIFFST ORE	ZCARD	TIME
EXPIRE	BITFIEL D	HINCRB Y	LPOP	SINTER	ZCOUNT	INFO
MOVE	DECR	HINCRB YFLOAT	LPUSH X	SINTERS TORE	ZDIFF	CLIENT KILL
PERSIST	DECRBY	HKEYS	LRANG E	SISMEM BER	ZDIFFSTORE	CLIENT LIST
PTTL	GET	HMGET	LREM	SMEMBE RS	ZINCRBY	CLIENT GETNAME
RANDO MKEY	GETRA NGE	HMSET	LSET	SMOVE	ZINTER	CLIENT SETNAME
RENAME	GETSET	HSET	LTRIM	SPOP	ZINTERSTOR E	CONFIG GET
RENAME NX	GETDEL	HSETN X	RPOP	SRAND MEMBE R	ZLEXCOUNT	MONITOR
SORT	GETEX	HVALS	LMOVE	SREM	ZMSCORE	SLOWLOG
TTL	INCR	HSCAN	RPOPL PUSH	SUNION	ZPOPMAX	ROLE

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
TYPE	INCRBY	HSTRLE N	RPUSH	SUNION STORE	ZPOPMIN	SWAPDB
SCAN	INCRBY FLOAT	HLEN	RPUSH X	SSCAN	ZRANDMEM BER	MEMORY
PEXPIRE AT	MGET	HRAND FIELD	LPUSH	SMISME MBER	ZRANGE	LASTSAVE
PEXPIRE	MSET	-	BLPOP	-	ZRANGEBYL EX	REPLCONF
OBJECT ENCODI NG	MSETN X	-	BRPOP	-	ZRANGEBYS CORE	LASTSAVE
OBJECT FREQ	PSETEX	-	BRPOP LPUSH	-	ZRANGESTO RE	COMMAN D
OBJECT IDLETIM E	SET	-	LPOS	-	ZRANK	COMMAN D COUNT
OBJECT REFCOU NT	SETBIT	-	-	-	ZREM	COMMAN D GETKEYS
RESTOR E	SETEX	-	-	-	ZREMRANGE BYLEX	COMMAN D INFO
TOUCH	SETNX	-	-	-	ZREMRANGE BYRANK	CONFIG
UNLINK	SETRAN GE	-	-	-	ZREMRANGE BYSCORE	-
EXPIREA T	STRLEN	-	-	-	ZREVRANGE	-
KEYS	SUBSTR	-	-	-	ZREVRANGE BYLEX	-
WAIT	-	-	-	-	ZREVRANGE BYSCORE	-
-	-	-	-	-	ZREVRANK	-
-	-	-	-	-	ZSCAN	-
-	-	-	-	-	ZSCORE	-
-	-	-	-	-	ZUNION	-
-	-	-	-	-	ZUNIONSTO RE	-

Table 8-30 Commands supported by DCS for Redis 6.0 enterprise edition (2)

HyperLo glog	Pub/Su b	Connecti on	Scripting	Geo	Stream	Bitmaps
PFADD	PSUBS CRIBE	AUTH	EVAL	GEOADD	XACK	BITCOU NT
PFCOUN T	PUBLIS H	CLIENT CACHIN G	EVALSHA	GEODIST	XADD	BITFIELD
PFMERG E	PUBSU B	CLIENT GETNAM E	SCRIPT DEBUG	GEOHASH	XAUTOCL AIM	BITFIELD _RO
PFSELFT EST	PUNSU BSCRIB E	CLIENT GETREDI R	SCRIPT EXISTS	GEOPOS	XCLAIM	ВІТОР
-	SUBSC RIBE	CLIENT ID	SCRIPT FLUSH	GEORADI US	XDEL	BITPOS
-	UNSUB SCRIBE	CLIENT INFO	SCRIPT KILL	GEORADI USBYMEM BER	XGROUP	GETBIT
-	-	CLIENT KILL	SCRIPT LOAD	GEORADI USBYMEM BER_RO	XINFO	SETBIT
-	-	CLIENT LIST	-	GEORADI US_RO	XLEN	-
-	-	CLIENT PAUSE	-	GEOSEARC H	XPENDIN G	-
-	-	CLIENT REPLY	-	GEOSEARC HSTORE	XRANGE	-
-	-	CLIENT SETNAM E	-	-	XREAD	-
-	-	CLIENT TRACKIN G	-	-	XREADGR OUP	-
-	-	CLIENT TRACKIN GINFO	-	-	XREVRAN GE	-

HyperLo glog	Pub/Su b	Connecti on	Scripting	Geo	Stream	Bitmaps
-	-	CLIENT UNBLOC K	-	-	XSETID	-
-	-	CLIENT UNPAUS E	-	-	XTRIM	-
-	-	ECHO	-	-	-	-
-	-	HELLO	-	-	-	-
-	-	PING	-	-	-	-
-	-	QUIT	-	-	-	-
-	-	RESET	-	-	-	-
-	-	SELECT	-	-	-	-

Commands Disabled by DCS for Redis 6.0

Table 8-31 Commands disabled in single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances

Generic (Key)	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG commands	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT- FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG- EPOCH

Generic (Key)	Server	Cluster	
-	-	CLUSTER RESET	

 Table 8-32 Redis commands disabled in Proxy Cluster DCS Redis 6.0 instances

Generic (Key)	Server	Connection
MIGRATE	BGREWRITEAOF	CLIENT CACHING
MOVE	BGSAVE	CLIENT GETREDIR
WAIT	CLIENT commands	CLIENT INFO
-	DEBUG OBJECT	CLIENT TRACKING
-	DEBUG SEGFAULT	CLIENT TRACKINGINFO
-	LASTSAVE	CLIENT UNPAUSE
-	PSYNC	RESET
-	SAVE	-
-	SHUTDOWN	-
-	SLAVEOF	-
-	LATENCY commands	-
-	MODULE commands	-
-	LOLWUT	-
-	SWAPDB	-
-	REPLICAOF	-
-	SYNC	-
-	ACL	-
-	FAILOVER	-

Table 8-33 Redis commands disabled in read/write splitting DCS Redis 6.0 instances

Generic	Server	Connection
MIGRATE	BGREWRITEAOF	CLIENT CACHING
WAIT	BGSAVE	CLIENT GETREDIR
-	DEBUG OBJECT	CLIENT INFO

Generic	Server	Connection
-	DEBUG SEGFAULT	CLIENT TRACKING
-	LASTSAVE	CLIENT TRACKINGINFO
-	LOLWUT	CLIENT UNPAUSE
-	MODULE LIST/LOAD/ UNLOAD	RESET
-	PSYNC	-
-	REPLICAOF	-
-	SAVE	-
-	SHUTDOWN [NOSAVE SAVE]	-
-	SLAVEOF	-
-	SWAPDB	-
-	SYNC	-
-	ACL	-
-	FAILOVER	-

Table 8-34 Redis commands disabled in DCS Redis 6.0 enterprise edition instances

Generic (Key)	Server	HyperLoglog
MIGRATE	SLAVEOF	PFDEBUG
-	SHUTDOWN	1
-	SAVE	-
-	BGSAVE	-
-	BGREWRITEAOF	-
-	SYNC	-
-	PSYNC	-
-	REPLICAOF	-

Commands That Can Be Renamed

Table 8-35 Commands that can be renamed

Command	command, keys, flushdb, flushall, hgetall, scan, hscan, sscan, and zscan For Proxy Cluster instances, the dbsize and dbstats commands can also be renamed.
Method	See Renaming Commands.

8.4 Commands Supported and Disabled in Web CLI

Web CLI is a command line tool provided on the DCS console. This section describes Web CLI's compatibility with Redis commands, including supported and disabled commands. Currently, only DCS for Redis 4.0 and later support Web CLI.

∩ NOTE

- Keys and values cannot contain spaces.
- If the value is empty, **nil** is returned after the **GET** command is executed.

Commands Supported in Web CLI

The following lists the commands supported when you use Web CLI. For details about the command syntax, visit the **Redis official website**. For example, to view details about the **SCAN** command, enter **SCAN** in the search box on **this page**.

Table 8-36 Commands supported by Web CLI (1)

Generic (Key)	String	List	Set	Sorted Set	Server
DEL	APPEND	RPUSH	SADD	ZADD	FLUSHALL
OBJECT	BITCOUN T	RPUSHX	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	BRPOPLR USH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	LINDEX	SDIFFSTO RE	ZINCRBY	TIME
MOVE	DECR	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	LLEN	SINTERST ORE	ZRANGEBYSCO RE	CLIENT KILL
PTTL	GET	LPOP	SISMEMB ER	ZRANK	CLIENT LIST

Generic (Key)	String	List	Set	Sorted Set	Server
RANDOM KEY	GETRAN GE	LPUSHX	SMEMBER S	ZREMRANGEB YRANK	CLIENT GETNAME
RENAME	GETSET	LRANGE	SMOVE	ZREMRANGEB YCORE	CLIENT SETNAME
RENAMEN X	INCR	LREM	SPOP	ZREVRANGE	CONFIG GET
SCAN	INCRBY	LSET	SRANDME MBER	ZREVRANGEBY SCORE	SLOWLOG
SORT	INCRBYFL OAT	LTRIM	SREM	ZREVRANK	ROLE
TTL	MGET	RPOP	SUNION	ZSCORE	SWAPDB
TYPE	MSET	RPOPLP U	SUNIONS TORE	ZUNIONSTORE	MEMORY
-	MSETNX	RPOPLP USH	SSCAN	ZINTERSTORE	-
-	PSETEX	-	-	ZSCAN	-
-	SET	-	-	ZRANGEBYLEX	-
-	SETBIT	-	-	ZLEXCOUNT	-
-	SETEX	-	-	-	-
-	SETNX	-	-	-	-
-	SETRANG E	-	-	-	-
-	STRLEN	-	-	-	-
-	BITFIELD	-	-	-	-

Table 8-37 Commands supported by Web CLI (2)

Hash	HyperLo glog	Connecti on	Scripting	Geo	Pub/Sub
HDEL	PFADD	AUTH	EVAL	GEOADD	UNSUBSCRIB E
HEXISTS	PFCOUN T	ECHO	EVALSHA	GEOHASH	PUBLISH
HGET	PFMERGE	PING	SCRIPT EXISTS	GEOPOS	PUBSUB

Hash	HyperLo glog	Connecti on	Scripting	Geo	Pub/Sub
HGETALL	-	QUIT	SCRIPT FLUSH	GEODIST	PUNSUBSCRI BE
HINCRBY	-	-	SCRIPT KILL	GEORADIUS	-
HINCRBYFL OAT	-	-	SCRIPT LOAD	GEORADIUS BYMEMBER	-
HKEYS	-	-	-	-	-
HMGET	-	-	-	-	-
HMSET	-	-	-	-	-
HSET	-	-	-	-	-
HSETNX	-	-	-	-	-
HVALS	-	-	-	-	-
HSCAN	-	-	-	-	-
HSTRLEN	-	-	-	-	-

Commands Disabled in Web CLI

The following lists the commands disabled when you use Web CLI.

Table 8-38 Commands disabled in Web CLI (1)

Generic (Key)	Server	Transactions	Cluster
MIGRATE	SLAVEOF	UNWATCH	CLUSTER MEET
WAIT	SHUTDOWN	REPLICAOF	CLUSTER FLUSHSLOTS
DUMP	DEBUG commands	DISCARD	CLUSTER ADDSLOTS
RESTORE	CONFIG SET	EXEC	CLUSTER DELSLOTS
WAITAOF	CONFIG REWRITE	MULTI	CLUSTER SETSLOT
-	CONFIG RESETSTAT	WATCH	CLUSTER BUMPEPOCH
-	SAVE	-	CLUSTER SAVECONFIG
-	BGSAVE	-	CLUSTER FORGET
-	BGREWRITEAOF	-	CLUSTER REPLICATE

Generic (Key)	Server	Transactions	Cluster
-	COMMAND	-	CLUSTER COUNT- FAILURE-REPORTS
-	KEYS	-	CLUSTER FAILOVER
-	MONITOR	-	CLUSTER SET-CONFIG- EPOCH
-	SYNC	-	CLUSTER RESET
-	PSYNC	-	-
-	ACL	-	-
-	MODULE	-	-
-	COMMAND COUNT	-	-
-	COMMAND DOCS	-	-
-	COMMAND GETKEYS	-	-
-	COMMAND GETKEYSANDFLAGS	-	-
-	COMMAND INFO	-	-
-	COMMAND LIST	-	-
-	RESTORE-ASKING	-	-

Table 8-39 Commands disabled in Web CLI (2)

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

8.5 Command Restrictions

Some Redis commands are supported by Redis Cluster instances for multi-key operations in the same slot. Restricted commands are listed in **Table 8-40**.

Some commands support multiple keys but do not support cross-slot access. For details, see **Table 8-42**. Restricted commands are listed in **Table 8-41**.

Table 8-43 lists commands restricted for read/write splitting instances.

□ NOTE

While running commands that take a long time to run, such as **FLUSHALL**, DCS instances may not respond to other commands and may change to the faulty state. After the command finishes executing, the instance will return to normal.

Redis Commands Restricted in Redis Cluster DCS Instances

Table 8-40 Redis commands restricted in Redis Cluster DCS instances

Category	Description		
Set			
SINTER	Returns the members of the set resulting from the intersection of all the given sets.		
SINTERSTORE	Equal to SINTER , but instead of returning the result set, it is stored in <i>destination</i> .		
SUNION	Returns the members of the set resulting from the union of all the given sets.		
SUNIONSTORE	Equal to SUNION , but instead of returning the result set, it is stored in <i>destination</i> .		
SDIFF	Returns the members of the set resulting from the difference between the first set and all the successive sets.		
SDIFFSTORE	Equal to SDIFF , but instead of returning the result set, it is stored in <i>destination</i> .		
SMOVE	Moves member from the set at source to the set at <i>destination</i> .		
Sorted Set			
ZUNIONSTORE	Computes the union of <i>numkeys</i> sorted sets given by the specified keys.		
ZINTERSTORE	Computes the intersection of <i>numkeys</i> sorted sets given by the specified keys.		
HyperLogLog	HyperLogLog		
PFCOUNT	Returns the approximated cardinality computed by the HyperLogLog data structure stored at the specified variable.		
PFMERGE	Merges multiple HyperLogLog values into a unique value.		

Category	Description	
Key		
RENAME	Renames key to newkey.	
RENAMENX	Renames key to newkey if newkey does not yet exist.	
BITOP	Performs a bitwise operation between multiple keys (containing string values) and stores the result in the destination key.	
RPOPLPUSH	Returns and removes the last element (tail) of the list stored at source, and pushes the element at the first element (head) of the list stored at <i>destination</i> .	
String	•	
MSETNX	Sets the given keys to their respective values.	

Redis Commands Restricted in Proxy Cluster DCS Instances

Table 8-41 Redis commands restricted in Proxy Cluster DCS instances

Category	Command	Restriction
Sets	SMOVE	For a Proxy Cluster instance, the source and destination keys must be in the same slot.
Sorted	BZPOPMAX	For a Proxy Cluster instance, all
sets	BZPOPMIN	keys transferred must be in the same slot.
		Not supported for Proxy Cluster DCS Redis 4.0 instances.
Geo	GEORADIUS	For a Proxy Cluster instance, all
	GEORADIUSBYMEMBER	keys transferred must be in the same slot.
	GEOSEARCHSTORE	 For a Proxy Cluster instance wi multiple databases, the STORE option is not supported.
Connectio n	CLIENT KILL	Only the following two formats are supported:
		- CLIENT KILL ip:port
		- CLIENT KILL ADDR ip:port
		 The id field has a random value, and it does not meet the idc1<idc2 li="" requirement.<="" tc1<tc2="" →=""> </idc2>

Category	Command	Restriction
	CLIENT LIST	Only the following two formats are supported: CLIENT LIST
		CLIENT LIST [TYPE normal master replica pubsub]
		 The id field has a random value, and it does not meet the idc1<idc2 li="" requirement.<="" tc1<tc2="" →=""> </idc2>
	SELECT index	Multi-DB of Proxy Cluster instances can be implemented by changing the keys. This solution is not recommended.
		For details about multi-DB restrictions on Proxy Cluster instances, see What Are the Constraints on Implementing Multiple Databases on a Proxy Cluster Instance?
HyperLogL	PFCOUNT	For a Proxy Cluster instance, all
og	PFMERGE	keys transferred must be in the same slot.
Keys	RENAME	For a Proxy Cluster instance, all
	RENAMENX	keys transferred must be in the same slot.

Category	Command	Restriction
	SCAN	 Proxy Cluster instances do not support the SCAN command in pipelines.
		• For a Proxy Cluster instance, you can issue the SCAN command to a specific shard by adding ip.port. (Shard IP addresses and ports can be queried through the icluster nodes command.) If the SCAN command issued on the specified shard does not return a desired response, your proxies may be an earlier version. In this case, contact customer service. icluster nodes xxx 192.168.00.00:1111@xxx xxx connected 10923-16383 xxx 192.168.00.01:2222@xxx xxx connected 0-5460 xxx 192.168.00.02:3333@xxx xxx connected 5461-10922 SCAN 0 match * COUNT 5 192.168.00.02:3333 1) "0" 2) 1) "key1" 2) "key2" 3) "key3" 4) "key4" 5) "key5"
Lists	BLPOP	For a Proxy Cluster instance, all
	BRPOP	keys transferred must be in the same slot.
	BRPOPLPUSH	
Pub/Sub	PSUBSCRIBE	Proxy Cluster instances do not support keyspace event subscription, so there would be no keyspace event subscription failure.
Scripting	EVAL	For a Proxy Cluster instance, all
	EVALSHA	keys transferred must be in the same slot.
		When the multi-DB function is enabled for a Proxy Cluster instance, the KEYS parameter will be modified. Pay attention to the KEYS parameter used in the Lua script.

Category	Command	Restriction
Server	MEMORY DOCTOR	For a Proxy Cluster instance, add
	MEMORY HELP	the <i>ip.port</i> of the node at the end of the command.
	MEMORY MALLOC-STATS	Do as follows to obtain the IP
	MEMORY PURGE	address and port number of a node (MEMORY USAGE is used as an
	MEMORY STATS	example): 1. Run the cluster keyslot <i>key</i>
	MEMORY USAGE	command to query the slot
	MONITOR	number of a key. 2. Run the icluster nodes command to query the IP address and port number corresponding to the slot where the key is. If the required information is not returned after you run the icluster nodes command, your Proxy Cluster instance may be of an earlier version. In this case, run the cluster nodes command. 3. Run the MEMORY USAGE key ip:port command. If multi-DB is enabled for the Proxy Cluster instance, run the MEMORY USAGE xxx:As {key} ip:port command, where xxx indicates the DB where the key value is. For example, DB0, DB1, and DB255 correspond to 000, 001, and 255, respectively. The following is an example for a single-DB Proxy Cluster instance: set key1 value1 OK get key1 value1 cluster keyslot key1 9189 icluster nodes xxx 192.168.00.00:1111@xxx xxx connected 10923-16383 xxx 192.168.00.00:3333@xxx xxx connected 5461-10922 MEMORY USAGE key1 192.168.00.02:3333

Category	Command	Restriction
Strings	ВІТОР	For a Proxy Cluster instance, all
	MSETNX keys transferred must be in same slot.	keys transferred must be in the same slot.
Transactio ns	WATCH	For a Proxy Cluster instance, all keys transferred must be in the same slot.
	MULTI	The order of cross-slot commands
	EXEC	in a transaction is not guaranteed. The following commands cannot be used in transactions: WATCH, MONITOR, RANDOMKEY, KEYS, SCAN, SUBSCRIBE, UNSUBSCRIBE, PSUBSCRIBE, PUNSUBSCRIBE, SCRIPT, EVAL, EVALSHA, DBSIZE, AUTH, FLUSHDB, FLUSHALL, CLIENT, MEMORY
Streams	XACK	Currently, Proxy Cluster instances
	XADD	do not support Streams.
	XCLAIM	
	XDEL	
	XGROUP	
	XINFO	
	XLEN	
	XPENDING	
	XRANGE	
	XTRIM	
	XREVRANGE	
	XREAD	
	XREADGROUP GROUP	
	XAUTOCLAIM	

Multi-Key Commands of Proxy Cluster Instances

Table 8-42 Multi-key commands of Proxy Cluster instances

Category	Command
Multi-key commands that support cross- slot access	DEL, MGET, MSET, EXISTS, SUNION, SINTER, SDIFF, SUNIONSTORE, SINTERSTORE, SDIFFSTORE, ZUNIONSTORE, ZINTERSTORE
Multi-key commands that do not support cross-slot access	SMOVE, SORT, BITOP, MSETNX, RENAME, RENAMENX, BLPOP, BRPOP, RPOPLPUSH, BRPOPLPUSH, PFMERGE, PFCOUNT, BLMOVE, COPY, GEOSEARCHSTORE, LMOVE, ZRANGESTORE

Redis Commands Restricted for Read/Write Splitting Instances

Table 8-43 Redis commands restricted for read/write splitting instances

Category	Command	Restriction
Connectio n	CLIENT KILL	 Only the following two formats are supported: CLIENT KILL ip:port CLIENT KILL ADDR ip:port The id field has a random value, and it does not meet the idc1<idc2 li="" requirement.<="" tc1<tc2="" →=""> </idc2>
	CLIENT LIST	 Only the following two formats are supported: CLIENT LIST CLIENT LIST [TYPE normal master replica pubsub] The id field has a random value, and it does not meet the idc1<idc2 li="" requirement.<="" tc1<tc2="" →=""> </idc2>

8.6 Other Command Usage Restrictions

This section describes restrictions on some Redis commands.

KEYS Command

In case of a large amount of cached data, running the **KEYS** command may block the execution of other commands for a long time or occupy exceptionally large memory. Therefore, when running the **KEYS** command, describe the exact pattern and do not use fuzzy **keys** *. Using **keys** * iterates all data, consuming CPU and

affecting service stability. Do not use the **KEYS** command in the production environment. Otherwise, the service running will be affected.

Commands in the Server Group

- While running commands that take a long time to run, such as FLUSHALL, DCS instances may not respond to other commands and may change to the faulty state. After the command finishes executing, the instance will return to normal.
- When the **FLUSHDB** or **FLUSHALL** command is run, execution of other service commands may be blocked for a long time in case of a large amount of cached data.
- You are not advised to run the **MONITOR** command in high-concurrency scenarios or during peak hours.

EVAL and EVALSHA Commands

- When the EVAL or EVALSHA command is run, at least one key must be contained in the command parameter. Otherwise, the error message "ERR eval/evalsha numkeys must be bigger than zero in redis cluster mode" is displayed.
- When the EVAL or EVALSHA command is run, a cluster DCS Redis instance
 uses the first key to compute slots. Ensure that the keys to be operated in
 your code are in the same slot. For details, visit the Redis official website.
- For the **EVAL** command:
 - Learn the Lua script features of Redis before running the EVAL command.
 For details, visit the Redis official website.
 - The execution timeout time of a Lua script is 5 seconds. Time-consuming statements such as long-time sleep and large loop statements should be avoided.
 - When calling a Lua script, do not use random functions to specify keys.
 Otherwise, the execution results are inconsistent on the master and standby nodes.

Debugging Lua Scripts

When you debug Lua scripts for Proxy Cluster and read/write splitting instances, only the asynchronous non-blocking mode **--ldb** is supported. The synchronous blocking mode **--ldb-sync-mode** is not supported. By default, the maximum concurrency on each proxy is **2**. This restriction does not apply to other instance types.

Other Restrictions

- The time limit for executing a Redis command is 15 seconds. To prevent other services from failing, a master/replica switchover will be triggered after the command execution times out.
- Cluster DCS Redis instances created before July 10, 2018 must be upgraded to support the following commands:
 - SINTER, SDIFF, SUNION, PFCOUNT, PFMERGE, SINTERSTORE, SUNIONSTORE, SDIFFSTORE, SMOVE, ZUNIONSTORE, ZINTERSTORE, EVAL, EVALSHA, BITOP,

RENAME, RENAMENX, RPOPLPUSH, MSETNX, SCRIPT LOAD, SCRIPT KILL, SCRIPT EXISTS, and SCRIPT FLUSH

8.7 Commands Supported and Disabled by DCS for Redis 3.0 (Discontinued)

DCS for Redis 3.0 is developed based on Redis 3.0.7 and is compatible with open-source protocols and commands. This section describes DCS for Redis 3.0's compatibility with Redis commands, including supported commands, disabled commands, unsupported scripts and commands of later Redis versions, and restrictions on command usage.

□ NOTE

DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 5.0 or later.

DCS Redis instances support most Redis commands. Any client compatible with the Redis protocol can access DCS.

- For security purposes, some Redis commands are disabled in DCS, as listed in Commands Disabled by DCS for Redis 3.0.
- Some Redis commands are supported by cluster DCS instances for multi-key operations in the same slot. For details, see **Command Restrictions**.
- Some Redis commands (such as KEYS, FLUSHDB, and FLUSHALL) have usage restrictions, which are described in Other Command Usage Restrictions.

Commands Supported by DCS for Redis 3.0

The following lists commands supported by DCS for Redis 3.0. For details about the command syntax, visit the **Redis official website**. For example, to view details about the **SCAN** command, enter **SCAN** in the search box on **this page**.

- Commands available since later Redis versions are not supported by earlier-version instances. Run a command on redis-cli to check whether it is supported by DCS for Redis. If the message "(error) ERR unknown command" is returned, the command is not supported.
- The following commands listed in the tables are not supported by Proxy Cluster instances:
 - List group: BLPOP, BRPOP, and BRPOPLRUSH
 - CLIENT commands in the Server group: CLIENT KILL, CLIENT GETNAME, CLIENT LIST, CLIENT SETNAME, CLIENT PAUSE, and CLIENT REPLY.
 - Server group: MONITOR
 - Transactions group: UNWATCH and WATCH
 - Key group: RANDOMKEY (for old instances)

Table 8-44 Commands supported by DCS Redis 3.0 instances (1)

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEN D	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOU NT	HEXIST S	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	ВІТОР	HGET	BRPOP LRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETAL L	LINDEX	SDIFFST ORE	ZINCRBY	TIME
MOVE	DECR	HINCRB Y	LINSER T	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRB YFLOAT	LLEN	SINTERS TORE	ZRANGEBYS CORE	CLIENT KILL
PTTL	GET	HKEYS	LPOP	SISMEM BER	ZRANK	CLIENT LIST
RANDO MKEY	GETRA NGE	HMGET	LPUSH X	SMEMBE RS	ZREMRANGE BYRANK	CLIENT GETNAME
RENAME	GETSET	HMSET	LRANG E	SMOVE	ZREMRANGE BYCORE	CLIENT SETNAME
RENAME NX	INCR	HSET	LREM	SPOP	ZREVRANGE	CONFIG GET
RESTOR E	INCRBY	HSETN X	LSET	SRAND MEMBE R	ZREVRANGE BYSCORE	MONITOR
SORT	INCRBY FLOAT	HVALS	LTRIM	SREM	ZREVRANK	SLOWLOG
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	ROLE
TYPE	MSET	-	RPOPL PU	SUNION STORE	ZUNIONSTO RE	-
SCAN	MSETN X	-	RPOPL PUSH	SSCAN	ZINTERSTOR E	-
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	-
KEYS	SET	-	RPUSH X	-	ZRANGEBYL EX	-
-	SETBIT	-	-	-	-	-
-	SETEX	-	-	-	-	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
-	SETNX	-	-	-	-	-
-	SETRAN GE	-	-	-	-	-
-	STRLEN	-	-	-	-	-

Table 8-45 Commands supported by DCS Redis 3.0 instances (2)

HyperLogl og	Pub/Sub	Transacti ons	Connecti on	Scripting	Geo
PFADD	PSUBSCRI BE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBS CRIBE	UNWATC H	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIB E	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSC RIBE	-	-	SCRIPT LOAD	GEORADIUSBY MEMBER

Commands Disabled by DCS for Redis 3.0

The following lists commands disabled by DCS for Redis 3.0.

Table 8-46 Redis commands disabled in single-node and master/standby DCS Redis 3.0 instances

Generic (Key)	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG commands
-	COMMAND
-	SAVE

Generic (Key)	Server
-	BGSAVE
-	BGREWRITEAOF

Table 8-47 Redis commands disabled in Proxy Cluster DCS Redis 3.0 instances

Generi c (Key)	Server	List	Transactio ns	Connecti on	Cluste r	codis
MIGRA TE	SLAVEOF	BLPOP	DISCARD	SELECT	CLUST ER	TIME
MOVE	SHUTDO WN	BRPOP	EXEC	-	-	SLOTSINF O
-	LASTSAVE	BRPOPL PUSH	MULTI	-	-	SLOTSDEL
-	DEBUG command s	-	UNWATCH	-	-	SLOTSMG RTSLOT
-	COMMAN D	-	WATCH	-	-	SLOTSMG RTONE
-	SAVE	-	-	-	-	SLOTSCHE CK
-	BGSAVE	-	-	-	-	SLOTSMG RTTAGSLO T
-	BGREWRIT EAOF	-	-	-	-	SLOTSMG RTTAGON E
-	SYNC	-	-	-	-	-
-	PSYNC	-	-	-	-	-
-	MONITOR	-	-	-	-	-
-	CLIENT command s	-	-	-	-	-
-	ОВЈЕСТ	-	-	-	-	-
-	ROLE	-	-	-		

8.8 Commands Supported and Disabled by DCS for Memcached (Discontinued)

Ⅲ NOTE

DCS for Memcached is no longer provided. You can use DCS Redis instances instead.

Memcached supports the TCP-based text protocol and binary protocol. Any clients compatible with a Memcached protocol can access DCS instances.

Memcached Text Protocol

The Memcached text protocol uses ASCII text to transfer commands, helping you compile clients and debug problems. DCS Memcached instances can even be directly connected using Telnet.

Compared with the Memcached binary protocol, the Memcached text protocol is compatible with more open-source clients, but the text protocol does not support authentication.

Clients can use the Memcached text protocol to access DCS Memcached instances only if password-free access is enabled. Password-free access means that access to DCS Memcached instances will not be username- and password-protected, and any Memcached clients that satisfy security group rules in the same VPC can access the instances. Enabling password-free access poses security risks. Exercise caution when enabling password-free access.

Table 8-48 lists the commands supported by the Memcached text protocol and describes whether these commands are supported by DCS Memcached instances.

Table 8-48 Commands supported by the Memcached text protocol

Command	Description	Supported by DCS
add	Adds data.	Yes
set	Sets data, including adding or modifying data.	Yes
replace	Replaces data.	Yes
append	Adds data after the value of the specified key.	Yes
prepend	Adds data before the value of a specified key.	Yes
cas	Checks and set data.	Yes
get	Queries data.	Yes
gets	Queries data details.	Yes

Command	Description	Supported by DCS
delete	Deletes data.	Yes
incr	Adds the specified amount to the requested counter.	Yes
decr	Removes the specified amount to the requested counter.	Yes
touch	Updates the expiration time of existing data.	Yes
quit	Closes the connection.	Yes
flush_all	Invalidates all existing data. NOTE The value of the delay option (if any) must be 0 .	Yes
version	Queries Memcached version information.	Yes
stats	Manages operation statistics. NOTE Currently, only basic statistics can be queried. Commands on optional parameters cannot be queried.	Yes
cache_memlimit	Adjusts the cache memory limit.	No
slabs	Queries usage of internal storage structures.	No
lru	Manages policies of deleting expired data.	No
lru_crawler	Manages threads of deleting expired data.	No
verbosity	Sets the verbosity level of the logging output.	No
watch	Inspects what's going on internally.	No

Memcached Binary Protocol

The Memcached binary protocol encodes commands and operations into specific structures before sending them. Commands are represented by predefined character strings.

The Memcached binary protocol provides more features but fewer clients than the Memcached text protocol. The Memcached binary protocol is more secure than the Memcached text protocol as it additionally supports simple authentication and security layer (SASL) authentication.

Table 8-49 lists the commands supported by the Memcached binary protocol and describes whether these commands are supported by DCS Memcached instances.

Table 8-49 Commands supported by the Memcached binary protocol

Comman d Code	Command	Description	Supported by DCS
0x00	GET	Queries data.	Yes
0x01	SET	Sets data, including adding or modifying data.	Yes
0x02	ADD	Adds data.	Yes
0x03	REPLACE	Replaces data.	Yes
0x04	DELETE	Deletes data.	Yes
0x05	INCREMENT	Adds the specified amount to the requested counter.	Yes
0x06	DECREMEN T	Removes the specified amount to the requested counter.	Yes
0x07	QUIT	Closes the connection.	Yes
0x08	FLUSH	Invalidates all existing data. NOTE The value of the delay option (if any) must be 0.	Yes
0x09	GETQ	Queries data. The client will not receive any response in case of failure.	Yes
0x0a	NOOP	No-operation instruction, equivalent to ping.	Yes
0x0b	VERSION	Queries Memcached version information.	Yes
0x0c	GETK	Queries data and adds a key into the response packet.	Yes
0x0d	GETKQ	Queries data and returns a key. The client will not receive any response in case of failure.	Yes
0x0e	APPEND	Adds data after the value of the specified key.	Yes
0x0f	PREPEND	Adds data before the value of a specified key.	Yes

Comman d Code	Command	Description	Supported by DCS
0x10	STAT	Queries statistics of DCS Memcached instances. NOTE Currently, only basic statistics can be queried. Commands on optional parameters cannot be queried.	Yes
0x11	SETQ	Sets data, including adding or modifying data. The SETQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x12	ADDQ	Adds data. Unlike the ADD command, the ADDQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x13	REPLACEQ	Replaces data. Unlike the REPLACE command, the REPLACEQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x14	DELETEQ	Deletes data. Unlike the DELETE command, the DELETEQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes

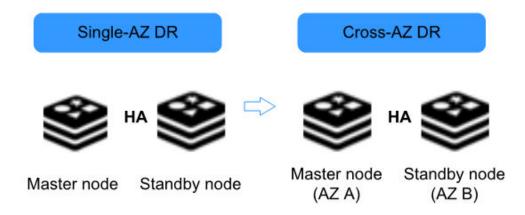
Comman d Code	Command	Description	Supported by DCS
0x15	INCREMENT Q	Adds the specified amount to the requested counter. Unlike the INCREMENT command, the INCREMENTQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x16	DECREMEN TQ	Removes the specified amount to the requested counter. Unlike the DECREMENT command, the DECREMENTQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x17	QUITQ	Closes the connection.	Yes
0x18	FLUSHQ	Clears data and returns no information. NOTE The value of the delay option (if any) must be 0 .	Yes
0x19	APPENDQ	Adds data after the value of the specified key. Unlike the APPEND command, the APPENDQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes
0x1a	PREPENDQ	Adds data before the value of a specified key. Unlike the PREPEND command, the PREPENDQ command only returns a response on failures. The client will not receive any response in the case of success.	Yes

Comman d Code	Command	Description	Supported by DCS
0x1c	TOUCH	Updates the expiration time of existing data.	Yes
0x1d	GAT	Queries data and updates the expiration time of existing data.	Yes
0x1e	GATQ	Queries data and returns a key. The client will not receive any response in case of failure.	Yes
0x23	GATK	Queries data, adds a key into the response packet, and updates the expiration time of existing data.	Yes
0x24	GATKQ	Queries data, returns a key, and updates the expiration time of existing data. The client will not receive any response in case of failure.	Yes
0x20	SASL_LIST_ MECHS	Asks the server what SASL authentication mechanisms it supports.	Yes
0x21	SASL_AUTH	Starts SASL authentication.	Yes
0x22	SASL_STEP	Further authentication steps are required.	Yes

9 Disaster Recovery and Multi-Active Solution

Whether you use DCS as the frontend cache or backend data store, DCS is always ready to ensure data reliability and service availability. The following figure shows the evolution of DCS DR architectures.

Figure 9-1 DCS DR architecture evolution



To meet the reliability requirements of your data and services, you can choose to deploy your DCS instance within a single AZ or across AZs.

Single-AZ HA Within a Region

Single-AZ deployment means deploying an instance within a physical equipment room. DCS provides process/service HA, data persistence, and hot standby DR policies for different types of DCS instances.

Single-node DCS instance: When DCS detects a process fault, a new process is started to ensure service HA.

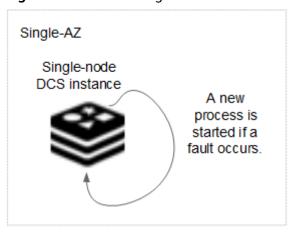


Figure 9-2 HA for a single-node DCS instance deployed within an AZ

Master/Standby, read/write splitting, or cluster DCS instance: By default, data is persisted to disk on the master node and incrementally synchronized and persisted to the standby node, achieving hot standby and data persistence.

The following figure shows the data synchronization and persistence of the master and standby node processes, including the processes of master/standby and read/write splitting instances and each shard of cluster instances.

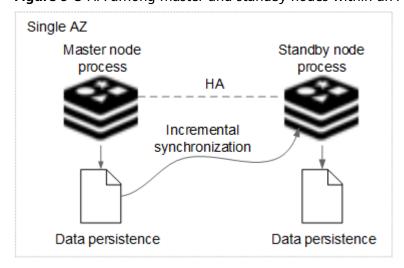


Figure 9-3 HA among master and standby nodes within an AZ

Cross-AZ DR Within a Region

The master and standby nodes of a DCS instance (single-node type not included) can be deployed across AZs (in different equipment rooms). Power supplies and networks of different AZs are physically isolated. When a fault occurs in the AZ where the master node is deployed, the standby node connects to the client and takes over data read and write operations.

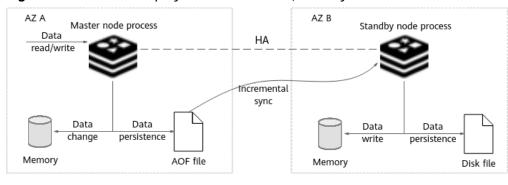
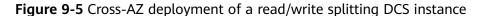
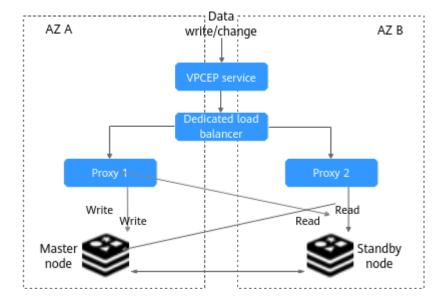


Figure 9-4 Cross-AZ deployment of a master/standby DCS instance





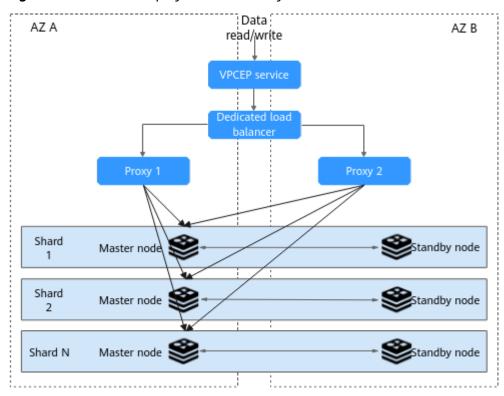
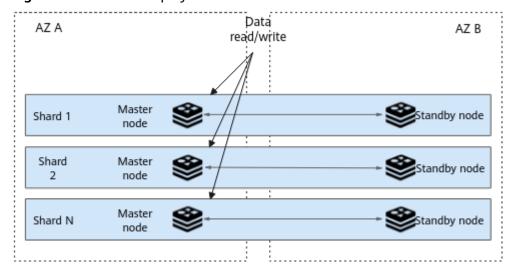


Figure 9-6 Cross-AZ deployment of a Proxy Cluster DCS instance

Figure 9-7 Cross-AZ deployment of a Redis Cluster DCS instance



When creating a master/standby, cluster, or read/write splitting DCS instance, select a standby AZ that is different from the master AZ as shown below.

Proxy Cluster Single-node Read/Write splitting Redis Cluster Instance Type Backup | Failover | Persistence 2 + ? Replicas AZ2 AZ3 AZ7 Primary AZ Standby AZ A73 A77 If your application has high availability requirements, deploy your instance across AZs to enhance fault tolerance.

Figure 9-8 Selecting different AZs

□ NOTE

You can also deploy your application across AZs to ensure both data reliability and service availability in the event of power supply or network disruptions.

Cross-Region Multi-Active

Currently, HUAWEI CLOUD DCS does not support cross-region multi-active because Redis does not have a mature active-active solution. **Active-active is different from disaster recovery or master/standby HA.**

Redis active-active across clouds or regions cannot be achieved because the customized Redis Serialization Protocols (RESP) are not unified. If active-active is required, it can be implemented through **dual-write on the application end**.

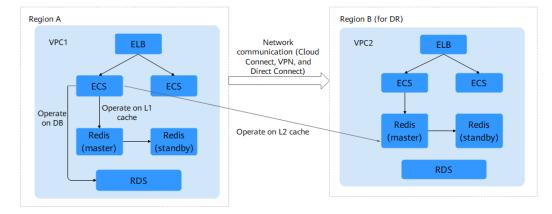


Figure 9-9 Dual-write on the application side to achieve multi-active

Note:

- The dual-write solution cannot ensure cache consistency (due to network problems). Applications need to tolerate cache inconsistency (by setting the time to live to achieve eventual consistency). If applications require strong cache consistency, this solution is not suitable. Currently, there is no solution in the industry to ensure strong cross-region cache consistency.
- 2. You are advised to perform operations on the cross-region L2 cache in asynchronous mode.

10 Comparing Versions and Specifications

10.1 Comparing Redis Instance Types

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

□ NOTE

- DCS for Redis 3.0 is no longer provided. You can use DCS for Redis 5.0 or later.
- 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 4.0 instance to Redis 5.0 or 6.0. 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.

Version

DCS supports Redis 3.0/4.0/5.0/6.0/7.0. **Table 10-1** describes the differences between these versions. For more information, see **Open Source Redis Versions**.

Table 10-1 Differences between Redis versions

Feature	Redis 3.0 (Discontinu ed)	Redis 4.0/5.0/6.0/7.0 Basic Edition	Redis 6.0 Enterprise Edition
Open- source compatib ility	3.x. For more information, see How Do I View the Version of a DCS Redis Instance?.	4.x, 5.x, 6.x, and 7.x. For more information, see How Do I View the Version of a DCS Redis Instance?.	6.x. For more information, see How Do I View the Version of a DCS Redis Instance?.

Feature	Redis 3.0 (Discontinu ed)	Redis 4.0/5.0/6.0/7.0 Basic Edition	Redis 6.0 Enterprise Edition
Time required for creating an instance	3–15 minutes Cluster instance: 10– 30 minutes	8 seconds	5 to 15 minutes
QPS	100,000 QPS per node	x86: 100,000 QPS per shard Arm: 80,000 QPS per shard	Enterprise (performance): 400,000 QPS per shard Enterprise (storage): 70,000 QPS per shard
Public network access	Supported	Public access can be enabled using ELB. For more information, see Enabling Public Access to Redis 4.0/5.0/6.0 and Obtaining the Access Addresses.	Public access can be enabled using ELB. For more information, see Enabling Public Access to Redis 4.0/5.0/6.0 and Obtaining the Access Addresses.
Domain name connecti on	Supported in VPC	Supported in VPC	Supported in VPC
Visualize d data manage ment	Not supported	Provides Web CLI for Redis access and data management.	Provides Web CLI for Redis access and data management.
Instance type	Single-node, master/ standby, and Proxy Cluster	Redis 4.0/5.0/6.0: single-node, master/ standby, read/write splitting, Proxy Cluster, and Redis Cluster Redis 7.0: single-node, master/standby, and Redis Cluster	Master/Standby
Capacity expansio n/ reduction	Online capacity expansion and reduction	Online capacity expansion and reduction	Online capacity expansion and reduction

Feature	Redis 3.0 (Discontinu ed)	Redis 4.0/5.0/6.0/7.0 Basic Edition	Redis 6.0 Enterprise Edition
Backup and restorati on	Supported for master/ standby and Proxy Cluster instances	Supported for master/ standby, Proxy Cluster, Redis Cluster, and read/ write splitting instances	Available for master/ standby instances.

Instance type

DCS provides single-node, master/standby, Proxy Cluster, Redis Cluster, and read/write splitting instance types. For details about their architectures and application scenarios, see **DCS Instance Types**.

10.2 Open Source Redis Versions

DCS for Redis is fully compatible with open-source Redis. This document describes the feature and compatibility updates of each Redis version.

Open Source Redis 7.2

New Features

See Valkey 7.2 release notes, Redis 7.2 release notes, and Redis 7.0 release notes.

- Introduced multi-part AOFs. A multi-part AOF splits an AOF into multiple small files, improving persistence reliability and recovery efficiency.
- Introduced Redis Functions. You can define and store functions through Lua scripts and the scripts can be reused later.
- Introduced new commands, such as ZMPOP and BZMPOP.
- Introduced sharded pub/sub for Redis Clusters to process Publish/Subscribe messages more efficiently.

Compatibility

- For details about breaking changes of open source Redis versions, see Valkey
 7.2 release notes, Redis 7.2 release notes, and Redis 7.0 release notes.
- User management is not supported.

Open Source Redis 6.2

New Features

See Redis 6.2 release notes and Redis 6.0 release notes.

- Added support for SSL and TLS encryption communication to ensure clientserver data transmission security.
- Introduced Redis Serialization Protocol 3 (RESP3). More data types and semantics, and RESP2 compatibility are available.

Introduced new commands, such as ZRANDMEMBER and COPY.

Compatibility

- For details about breaking changes of open source Redis versions, see Redis
 6.2 release notes and Redis 6.0 release notes.
- Supported SSL.
- Compatible with RESP2 and RESP3.
- Supported user management which is different from open-source ACL. User read-only and read/write permissions are controlled. The username and password are specified in the *username*:password or *username* password format. For details, see Configuring DCS Redis ACL Users.

Open Source Redis 5.0

New Features

See Redis 5.0 release notes.

- Introduced the stream type for message streams and log data.
- Optimized the RDB file format. The size and loading time of persistence files are reduced.
- Introduced new commands and options, such as XCLAIM, XINFO, and XTRIM.

Compatibility

For details about breaking changes of open source Redis versions, see **Redis 5.0** release notes.

Open Source Redis 4.0

New Features

See Redis 4.0 release notes.

- Introduced modules. Developers can extend Redis through modules instead of modifying Redis source code.
- Optimized the replication protocol. Supported partial re-synchronization. Data synchronization can be more efficient even after master and replica servers are disconnected.
- Added the least frequently used (LFU) policy. Redis can evict cache data by key usage frequency, in addition to the least recently used (LRU) policy.
- Introduced the hybrid persistence mode. An RDB snapshot can be embedded in an AOF, ensuring data security in faster recovery.
- Introduced new commands and options, such as **MEMORY** and **SWAPDB**.

Compatibility

- For details about breaking changes of open source Redis versions, see Redis
 4.0 release notes.
- SSL is not supported.
- RESP2 is supported.
- Supported user management. User read-only and read/write permissions are controlled. The username and password are specified in the

username: password format. For details, see Configuring DCS Redis ACL Users.

10.3 Comparing Enterprise and Basic Editions

Huawei Cloud DCS enterprise edition is completely developed by Huawei Cloud, and compatible with open-source Redis software. This edition uses the multi-thread master-N*worker model instead of the conventional master-worker model. Each worker thread can listen on ports to monitor requests for establishing network connections, accept the requests and establish network connections, read and write data (in network connections, for example, sockets), and parse and process Redis commands. As a result, performance is improved by N times.

Table 10-2 Comparing enterprise and basic editions

Item	Basic Edition	Enterprise Editions
Open-source compatibility	Open-source Redis 4.0/5.0/6.0/7.0, single- threaded	Compatible with open-source Redis 6.0, multi-threaded
Performance	x86: 100,000 QPS per shard Arm: 80,000 QPS per shard Up to 1 ms latency	Enterprise (performance): 400,000 QPS per shard Enterprise (storage): 70,000 QPS per shard Up to 1 ms latency
Instance specifications	Instance types: single- node, master/standby, cluster, and read/write splitting • Single-node and master/standby: 128 MB to 64 GB • Read/write splitting: 8- 32 GB • Cluster: up to 2048 GB	Currently, only the master/ standby instance type is available. There are two editions: • Enterprise (performance): 8– 64 GB • Enterprise (storage): 8–32 GB memory, and up to 256 GB with SSD storage For details about the differences between enterprise (performance) and enterprise (storage) editions, see Table 10-3.

Item	Basic Edition	Enterprise Editions
Data security	 Fine-grained authorization and IP address whitelists Data persistence and backup (except for the single-node type) Cross-AZ DR Automatic failover Capacity expansion and instance type change with a few clicks 	 Fine-grained authorization and security groups Persistence and backup Cross-AZ DR Automatic failover Capacity expansion with a few clicks

Table 10-3 Differences between enterprise (performance) and enterprise (storage) editions

Item	Enterprise (Performance) Edition	Enterprise (Storage) Edition
QPS	400,000 QPS per shard, higher than the storage edition	70,000 QPS per shard
Storage	SSD storage is not supported. Data is stored entirely in memory. Data is persisted using AOF files.	Memory + SSD Memory caches hot data and SSDs store all data.
Backup file format	AOF and RDB	RDB

Comparing DCS and Open-Source Cache Services

DCS supports single-node, master/standby, and cluster instances, ensuring high read/write performance and fast data access. It also supports various instance management operations to facilitate your O&M. With DCS, you only need to focus on the service logic, without concerning about the deployment, monitoring, scaling, security, and fault recovery issues.

DCS is compatible with open-source Redis and Memcached, and can be customized based on your requirements. This renders DCS unique features in addition to the advantages of open-source cache databases.

DCS for Redis vs. Open-Source Redis

Table 11-1 Differences between DCS for Redis and open-source Redis

Feature	Open-Source Redis	DCS for Redis
Service deployme nt	Requires 0.5 to 2 days to prepare servers.	 V6.0 enterprise edition: 5 to 15 minutes. V4.0 and later basic edition: 15 to 60 seconds.
Version	-	Deeply engaged in the open-source community and supports the latest Redis version. Currently, Redis 4.0, 5.0, 6.0, and 7.0 are supported.
Security	Network and server safety is the user's responsibility.	 Network security is ensured using HUAWEI CLOUD VPCs and security groups. Data reliability is ensured by data replication and scheduled backup.

Feature	Open-Source Redis	DCS for Redis	
Performa nce	-	x86-based: 100,000 QPS per shard; Arm-based: 80,000 QPS per shard.	
		QPS per shard for a Redis 6.0 enterprise (performance) instance node: 400,000/s; QPS per shard for an enterprise (storage) instance node: 70,000/s.	
Monitorin g	Provides only basic statistics.	Provides more than 30 monitoring metrics and customizable alarm threshold and policies. • Various metrics	
		 External metrics include the number of commands, concurrent operations, connections, clients, and denied connections. 	
		 Resource usage metrics include CPU usage, physical memory usage, network input throughput, and network output throughput. 	
		 Internal metrics include instance capacity usage, as well as the number of keys, expired keys, PubSub channels, PubSub patterns, keyspace hits, and keyspace misses. 	
		 Custom alarm thresholds and policies for different metrics to help identify service faults. 	
Backup and	Supported	Supports scheduled and manual backup. Backup files can be downloaded.	
restoratio n		Backup data can be restored on the console.	
Paramete r	No visualized parameter	Visualized parameter management is supported on the console.	
managem ent	management	Configuration parameters can be modified online.	
		Data can be accessed and modified on the console.	

Feature	Open-Source Redis	DCS for Redis
Scale-up	Interrupts services and involves a complex procedure, from modifying the server RAM to modifying Redis memory and restarting the OS and services.	 Supports online scale-up and scale-down without interrupting services. Specifications can be scaled up or down within the available range based on service requirements.
O&M	Manual O&M	24/7 end-to-end O&M services

DCS for Memcached vs. Open-Source Memcached

Table 11-2 Differences between DCS for Memcached and open-source Memcached

Feature	Open-Source Memcached	DCS for Memcached
Service deployme nt	Requires 0.5 to 2 days to prepare servers.	Creates an instance in 5 to 15 minutes.
Security	Network and server safety is the user's responsibility.	 Network security is ensured using HUAWEI CLOUD VPCs and security groups. Data reliability is ensured by data replication and scheduled backup.
Performa nce	-	100,000 QPS per node

Feature	Open-Source Memcached	DCS for Memcached
Monitorin g	Provides only basic statistics.	Provides more than 30 monitoring metrics and customizable alarm threshold and policies.
		Various metrics
		 External metrics include the number of commands, concurrent operations, connections, clients, and denied connections.
		 Resource usage metrics include CPU usage, physical memory usage, network input throughput, and network output throughput.
		 Internal metrics include instance capacity usage, as well as the number of keys, expired keys, PubSub channels, PubSub patterns, keyspace hits, and keyspace misses.
		Custom alarm thresholds and policies for different metrics to help identify service faults.
Backup and restoratio n	Not supported	 Supports scheduled and manual backup. Backup data can be restored on the console.
Visualized maintena nce	No visualized parameter	Visualized parameter management is supported on the console.
Tice	management	 Configuration parameters can be modified online.
Scale-up	Interrupts services and involves a	 Supports online scale-up without interrupting services.
	complex procedure, from modifying the server RAM to modifying Redis memory and restarting the OS and services.	Specifications can be scaled up or down within the available range based on service requirements.
O&M	Manual O&M	24/7 end-to-end O&M services
Data persistenc e	Not supported	Supported for master/standby instances

12 Notes and Constraints

Redis Instance

Table 12-1 Notes and constraints

Item	Notes and Constraints
Instance version	Currently, DCS supports Redis 3.0 (discontinued), 4.0, 5.0, 6.0, and 7.0. Redis instances cannot be upgraded, but data of an earlier instance can be migrated to a later one.
Data security	 To control access to DCS Redis 3.0 and Redis 6.0 enterprise edition instances, you can use security groups. IP whitelists are not supported. To control access to DCS Redis 4.0 and later basic edition instances, you can use IP whitelists. Security groups are not supported. DCS Redis 6.0/7.0 basic edition supports SSL encryption.
Data persistence	 Not available for single-node instances. Master/Standby, read/write splitting, and cluster (except single-replica cluster) instances: Data persistence is supported by default.
Read/Write splitting	 Read/Write splitting instances: Read/Write splitting is implemented on the server by default. Redis Cluster and master/standby instances: Read/Write splitting is implemented on the client, which requires manual configurations. Read/Write splitting is not supported for other instances.
Data backup	Automatic or manual data backup is supported on the console for instances other than single-node ones.
VPC and subnet	Fixed once the instance is created.

Instance Change

Table 12-2 Notes and constraints

Item	Notes and Constraints		
Changing Redis instance specifications	You are advised to change instances during off-peak hours. Otherwise, the change may fail.		
or types	Change the replica quantity and capacity separately.		
	Only one replica can be deleted per operation.		
	 For more information, see Modifying DCS Instance Specifications. 		
Adjusting DCS instance bandwidth	This function is unavailable for enterprise edition DCS Redis instances.		
	This function is available only for instances in the Running state.		
	The bandwidth adjustment range is from the instance's assured bandwidth to its maximum bandwidth. Generally, the maximum bandwidth per shard is 2048 Mbit/s when the physical machine of the instance node has sufficient resources.		
Changing cluster instances to be across	Available only for single-AZ cluster instances with two or more replicas.		
AZs	To upgrade AZs for Proxy Cluster instances:		
	 Service running may fluctuate during the change. Perform this operation during off-peak hours. 		
	 If your application cannot reconnect or handle exceptions, try restarting the application after the change. 		
	To upgrade AZs for Redis Cluster instances:		
	 Changing AZs will not interrupt services or the master node, but will slightly affect performance. Perform this operation during off-peak hours. 		
	 Changing AZs interrupts connections to some replicas. Ensure your application can automatically recover from exceptions and reconnect to Redis. 		

Data Migration

Table 12-3 Notes and constraints

Item	Notes and Constraints	
Version upgrade	To migrate an instance, the target instance version must be later than the source one. Migrating a later instance to an earlier one may fail.	
Online migration	To migrate Redis instances online on the DCS console, the network between the source and target must be connected and the SYNC and PSYNC commands must be allowed on the source.	
	 You cannot use public networks for online migration. 	
	The source must be Redis 3.0 or later.	
	 You are advised to perform online migration during off-peak hours. Otherwise, the CPU usage of the source instance may surge and the latency may increase. 	
IP switch	The IP addresses and domain names of the source and target DCS Redis instances can be switched on the console after the source instance is fully and incrementally migrated.	
	Unavailable for enterprise edition instances.	
	Unavailable for Redis Cluster instances.	

13 Billing

DCS supports the pay-per-use mode and yearly/monthly billing mode. For details, see **Product Pricing Details**.

□ NOTE

The billing modes available on the console vary by region. Some regions do not support the yearly/monthly billing mode.

Billing Items

DCS usage is billed by DCS instance specification.

Billing Item	Description
DCS instance	Billing based on DCS instance specifications.

Note: HUAWEI CLOUD DCS charges based on the selected DCS instance specifications instead of the actual cache capacity.

Billing Modes

DCS provides two billing modes: pay-per-use and yearly/monthly. Pay-per-use is recommended if you are unsure of your future service needs and want to avoid paying for unused resources. However, if you are sure of your needs, yearly/monthly will be less expensive.

- Yearly/Monthly: Provides a larger discount than pay-per-use mode and is recommended for long-term users.
- Pay-per-use (hourly): You can start and stop DCS instances as needed and will be billed based on the duration of your use of DCS instances. Billing starts when a DCS instance is created and ends when the DCS instance is deleted. The minimum time unit is one second.
- You can switch between the yearly/monthly and pay-per-use modes.

Configuration Changes

You can change the specifications of a DCS Redis or Memcached instance, that is, scale up or down an instance and change the instance type. After you successfully change the specifications, the instance is billed based on new specifications. For details, see **Modifying DCS Instance Specifications**.

Renewal

You can renew a resource package upon its expiration, or you can set autorenewal rules for a resource package. For more information about renewing resource packages, see **Renewal Management**.

FAQ

For more information about DCS billing, see the **Purchasing and Billing FAQs**.

14 Permissions Management

If you need to assign different permissions to employees in your enterprise to access your DCS resources, IAM is a good choice for fine-grained permissions management. IAM provides identity authentication, permissions management, and access control, helping you secure access to your Huawei Cloud resources.

With IAM, you can use your Huawei Cloud account to create IAM users for your employees, and assign permissions to the users to control their access to specific resource types. For example, some software developers in your enterprise need to use DCS resources but should not be allowed to delete DCS instances or perform any other high-risk operations. In this scenario, you can create IAM users for the software developers and grant them only the permissions required for using DCS resources.

If your Huawei Cloud account does not require individual IAM users for permissions management, skip this section.

IAM can be used free of charge. You pay only for the resources in your account. For more information about IAM, see the IAM Service Overview.

DCS Permissions

By default, new IAM users do not have permissions assigned. You need to add a user to one or more groups, and attach permissions policies or roles to these groups. Users inherit permissions from the groups to which they are added and can perform specified operations on cloud services based on the permissions.

DCS is a project-level service deployed and accessed in specific physical regions. To assign DCS permissions to a user group, specify the scope as region-specific (for example, CN-Hong Kong) projects and select projects for the permissions to take effect. If **All projects** is selected, the permissions will take effect for the user group in all region-specific projects. When accessing DCS, the users need to switch to a region where they have been authorized to use this service.

You can grant users permissions by using roles and policies.

Roles: A type of coarse-grained authorization mechanism that defines
permissions related to user responsibilities. This mechanism provides only a
limited number of service-level roles for authorization. When using roles to
grant permissions, you need to also assign other roles on which the
permissions depend to take effect. However, roles are not an ideal choice for
fine-grained authorization and secure access control.

Policies: A type of fine-grained authorization mechanism that defines
permissions required to perform operations on specific cloud resources under
certain conditions. This mechanism allows for more flexible policy-based
authorization, meeting requirements for secure access control. For example,
you can grant DCS users only the permissions for operating DCS instances.
Most policies define permissions based on APIs. For the API actions supported
by DCS, see Permissions Policies and Supported Actions.

Table 1 lists all the system-defined roles and policies supported by DCS.

Table 14-1 System-defined roles and policies supported by DCS

Role/Policy Name	Description	Туре	Dependency
DCS FullAccess	All permissions for DCS. Users granted these permissions can operate and use all DCS instances.	System- defined policy	Actions required for buying professional instances: iam:permissions :listRolesForAge ncyOnProject iam:agencies:lis tAgenciesiam:ro les:listRoles iam:permissions :grantRoleToAg encyOnProject iam:agencies:cr eateAgency iam:agencies:de leteAgency
DCS UserAccess	Common user permissions for DCS, excluding permissions for creating, modifying, deleting DCS instances and modifying instance specifications.	System- defined policy	None
DCS ReadOnlyAcces s	Read-only permissions for DCS. Users granted these permissions can only view DCS instance data.	System- defined policy	None
DCS Administrator	Administrator permissions for DCS. Users granted these permissions can operate and use all DCS instances.	System- defined role	The Server Administrator and Tenant Guest roles need to be assigned in the same project.

Role/Policy Name	Description	Туре	Dependency
DCS AgencyAccess	Permissions to assign to DCS agencies. These permissions are used by a tenant to delegate DCS to perform the following operations on tenant resources when necessary. They are irrelevant to the operations performed by authorized users. • Querying a subnet • Querying the subnet list • Querying a port • Querying the port list • Updating a port	System- defined policy	None
	Creating a port		

□ NOTE

The **DCS UserAccess** policy is different from the **DCS FullAccess** policy. If you configure both of them, you cannot create, modify, delete, or scale DCS instances because deny statements will take precedence over allowed statements.

Table 2 lists the common operations supported by each system policy of DCS. Please choose proper system policies according to this table.

Table 14-2 Common operations supported by each system policy

Operation	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAcce ss	DCS Administrat or
Modifying instance configurati on parameters	√	✓	×	✓
Deleting backgroun d tasks	√	√	×	√
Accessing instances using Web CLI	✓	√	×	√

Operation	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAcce ss	DCS Administrat or
Modifying instance running status	✓	✓	×	√
Expanding instance capacity	√	×	×	√
Changing instance passwords	√	√	×	√
Modifying DCS instances	√	×	×	√
Performing a master/ standby switchover	√	✓	×	√
Backing up instance data	√	√	×	√
Analyzing big keys or hot keys	√	√	×	√
Creating DCS instances	√	×	×	√
Deleting instance backup files	√	✓	×	√
Restoring instance data	√	✓	×	√
Resetting instance passwords	√	√	×	√
Migrating instance data	√	√	×	√

Operation	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAcce ss	DCS Administrat or
Downloadi ng instance backup data	✓	✓	×	√
Deleting DCS instances	√	×	×	√
Querying instance configurati on parameters	✓	✓	✓	✓
Querying instance restoration logs	√	√	√	√
Querying instance backup logs	✓	✓	✓	√
Querying DCS instances	√	√	√	√
Querying instance backgroun d tasks	√	✓	√	✓
Querying all instances	√	√	√	√
Operating slow queries	√	√	√	√

Helpful Links

- IAM Service Overview
- Creating a User and Granting DCS Permissions
- Permissions Policies and Supported Actions

15 Basic Concepts

DCS Instance

An instance is the minimum resource unit provided by DCS.

You can select the Redis or Memcached cache engine. Instance types can be single-node, master/standby, or cluster. For each instance type, multiple specifications are available.

For details, see DCS Instance Specifications and DCS Instance Types.

Public Network Access

A Redis 3.0 instance with an Elastic IP (EIP) can be accessed on a client using the EIP. A Redis 4.0/5.0/6.0 instance can be accessed in public using Elastic Load Balance (ELB).

Stunnel is used to encrypt communication content in public network access. The network delay is slightly higher than that in the VPC, so public network access is suitable for local commissioning in the development phase.

For details, see the **public access instructions**.

Password-Free Access

DCS instances can be accessed in a VPC without passwords. Latency is lower because no password authentication is involved.

You can enable password-free access for instances that do not have sensitive data. To ensure data security, you are not allowed to enable password-free access for instances enabled with public network access.

For details, see Configuring the Redis Password (Modifying the Redis Instance Access Mode).

Maintenance Time Window

The maintenance time window is the period when the DCS service team upgrade and maintain the instance.

DCS instance maintenance takes place only once a quarter and does not interrupt services. Even so, you are advised to select a time period when the service demand is low.

When creating an instance, you must specify a maintenance time window, which can be modified after the instance is created.

For details, see Viewing Instance Details.

Cross-AZ Deployment

Master/Standby, cluster, and read/write splitting instances are deployed across different AZs with physically isolated power supplies and networks Applications can also be deployed across AZs to achieve HA for both data and applications.

When creating an instance, you can select a primary AZ and a standby AZ.

Shard

A **shard** is a management unit of a cluster DCS Redis instance. Each shard corresponds to a redis-server process. A cluster consists of multiple shards. Each shard has multiple slots. Data is distributed to the slots. The use of shards increases cache capacity and concurrent connections.

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

Replica

A replica is a **node** of a DCS instance. A single-replica instance has no standby node. A two-replica instance has one master node and one standby node. By default, each master/standby instance has two replicas. If the number of replicas is set to three for a master/standby instance, the instance has one master node and two standby nodes. A single-node instance has only one node.

16 Related Services

DCS is used together with other HUAWEI CLOUD services, including VPC, ECS, IAM, Cloud Eye, CTS, and Object Storage Service (OBS).

OBS

ECS

Accessing an instance through a client

User permissions management

Operation logs audit

Figure 16-1 Relationships between DCS and other services

VPC

IAM

A VPC is an isolated virtual network environment on HUAWEI CLOUD. You can configure IP address ranges, subnets, and security groups, assign EIPs, and allocate bandwidth in a VPC.

DCS

DCS runs in VPCs. The VPC service manages EIPs and bandwidth, and provides security groups. You can configure access rules for security groups to secure the access to DCS.

ECS

An ECS is a cloud server that provides scalable, on-demand computing resources for secure, flexible, and efficient applications.

CTS

You can access and manage your DCS instances using an ECS.

IAM

IAM provides identity authentication, permissions management, and access control.

With IAM, you can control access to DCS.

Cloud Eye

Cloud Eye is a secure, scalable, and integrated monitoring service. With Cloud Eye, you can monitor your DCS service and configure alarm rules and notifications.

Cloud Trace Service (CTS)

CTS provides you with a history of operations performed on cloud service resources. With CTS, you can query, audit, and backtrack operations. The traces include the operation requests sent using the management console or open APIs and the results of these requests.

OBS

OBS provides secure, cost-effective storage service using objects as storage units. With OBS, you can store and manage the lifecycle of massive amounts of data.

You can store DCS instance backup files in OBS.