# Distributed Cache Service

# Service Overview

**Issue** 01

**Date** 2024-12-10

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

### Trademarks and Permissions


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

### Notice

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

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

## Huawei Cloud Computing Technologies Co., Ltd.

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

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

# Contents



# 1 What Is DCS?

Huawei Cloud Distributed Cache Service (DCS) is an online, distributed, fast in-memory 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.

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

**Table 1-1** DCS for Redis 4.0/5.0/6.0

| | |
|---|---|
| Instance type | DCS for Redis provides the following types of instances to suit different service scenarios:<br>• 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.<br>• 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.<br>• 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.<br>• 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.<br>• 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 specifications | DCS for Redis provides instances of different specifications, ranging from 128 MB to 1024 GB. |

| | |
|---|---|
| Open-source compatibility | DCS instances are compatible with open-source Redis 4.0/5.0/6.0. |
| Underlying architecture | Containerized deployment on physical machines. 100,000 QPS at a single node for a Redis 4.0/5.0 instance, 150,000 QPS for a single-node Redis 6.0 basic edition instance, and 170,000 QPS for a master/standby Redis 6.0 basic edition instance. |
| High availability (HA) and disaster recovery (DR) | All instances except single-node ones can be deployed across AZs within a region with physically isolated power supplies and networks. |

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

# 2 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 highest 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.

# 3 Security

## 3.1 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**.

## 3.2 Data Protection

DCS takes different measures to keep data secure and reliable.

**Table 3-1** 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 (intra-city DR). | **Disaster Recovery and Multi-Active Solution** |

| Measure | Description | Reference |
| --- | --- | --- |
| 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** |

# 3.3 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.

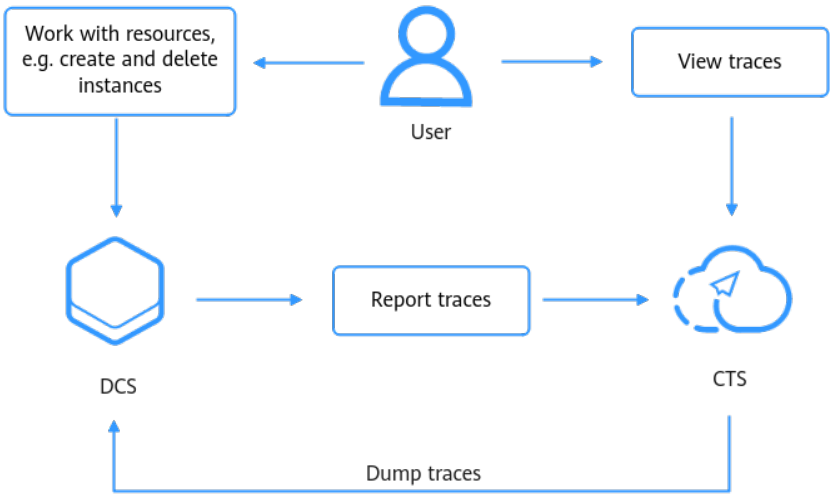
After you enable CTS and configure a tracker, CTS can record management and data traces of DCS for auditing.

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 3-1** CTS




# 3.4 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 3-2** 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. |
| 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. |

# 3.5 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**.

# 3.6 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.

## 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 whitelists.

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

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 read-heavy, 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.

# 4 DCS Instance Types

## 4.1 Single-Node Redis

Each single-node DCS Redis instance has only one node and does not support data persistence. They are suitable for cache services that do not require data reliability.

**NOTE**

- 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.
- Single-node instances cannot ensure data persistence and do not support automatic or manual data backup. Exercise caution when using them.

### 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 Redis 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.

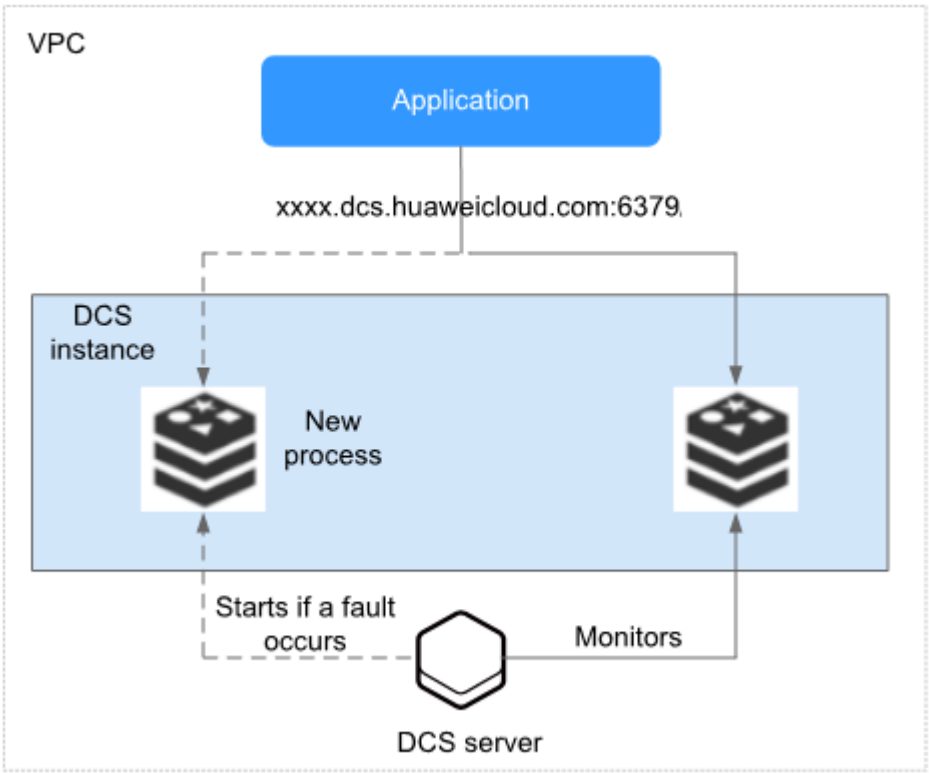
## Architecture

Figure 4-1 shows the architecture of a single-node DCS Redis instance.

NOTE

For Redis 4.0 and later, 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.

**Figure 4-1** 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 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 single-node DCS instance, which has only one node and one Redis process.

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

# 4.2 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 master/standby 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.

   NOTE

   - 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 **Access in Different Languages**.

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**

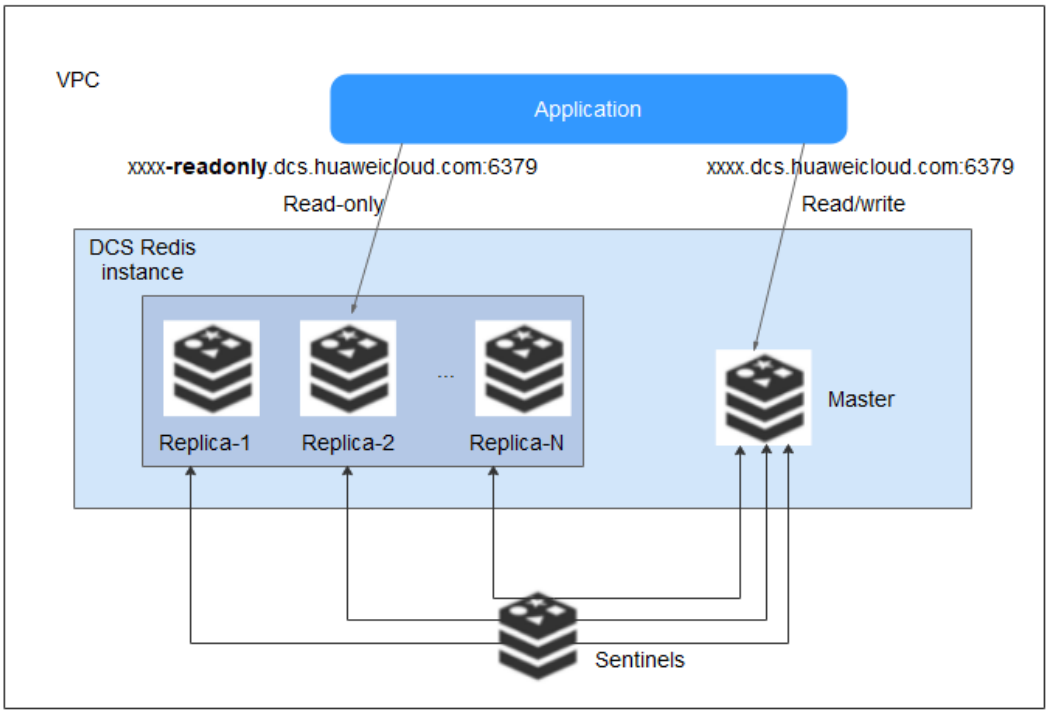
   Master/standby DCS Redis 4.0/5.0/6.0 instances support client read/write splitting. When connecting to such an instance, you can use the read/write address to connect to the master node or use the read-only address to connect to the standby node.

   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 4.0/5.0/6.0 Instances

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

**Figure 4-2** Architecture of a master/standby DCS Redis 4.0/5.0/6.0 instance




Architecture description:

1. Each master/standby DCS Redis 4.0/5.0/6.0 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 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 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 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) is faulty. For higher reliability and lower latency, use read/write splitting instances.

# 4.3 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.

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 4.0/5.0/6.0 Instances

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 4-1** 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.

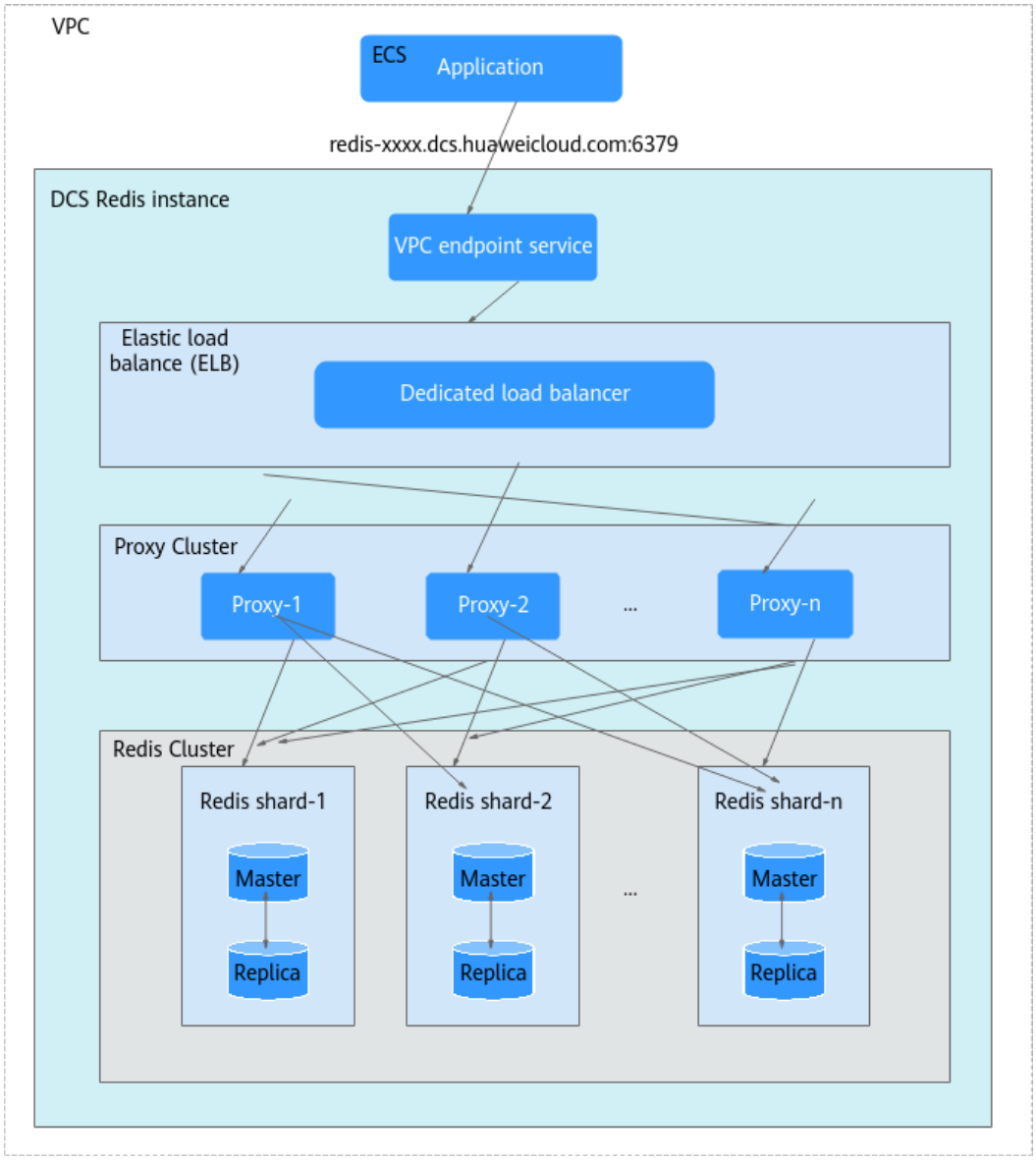
  

**Table 4-1** Total memory, proxies, and shards of Proxy Cluster DCS Redis 4.0/5.0/6.0 instances

| 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 |
| 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 4-3** Proxy Cluster DCS Redis 4.0/5.0/6.0 instances




Architecture description:

- **VPC**

  The VPC where all nodes of the instance are run.

  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 **Access in Different Languages**.

- **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.

# 4.4 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.

NOTE

- 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 **Access in Different Languages**.

## 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 4-2** 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.

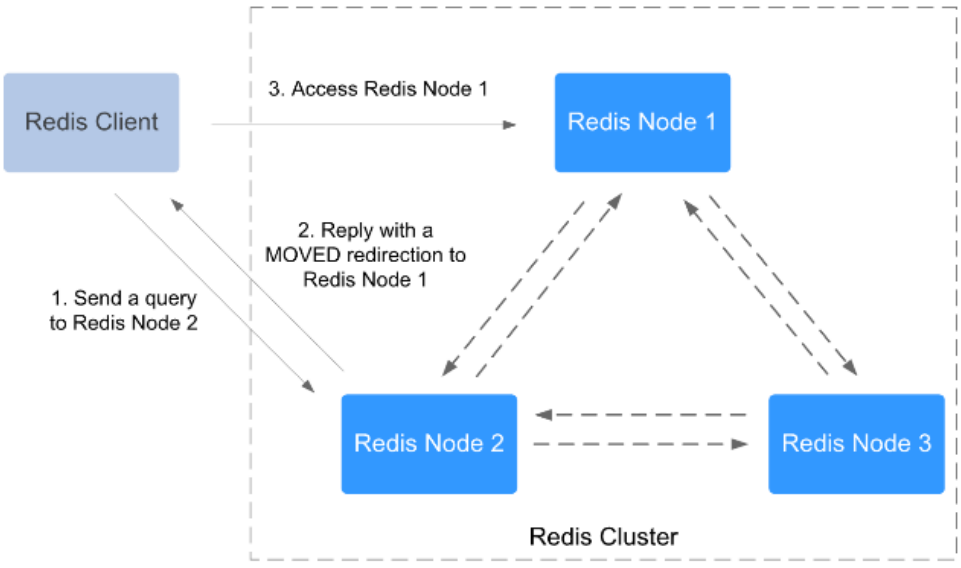
**Table 4-2** Specifications of Redis Cluster DCS instances

| 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 4-4** Distributed architecture of Redis Cluster

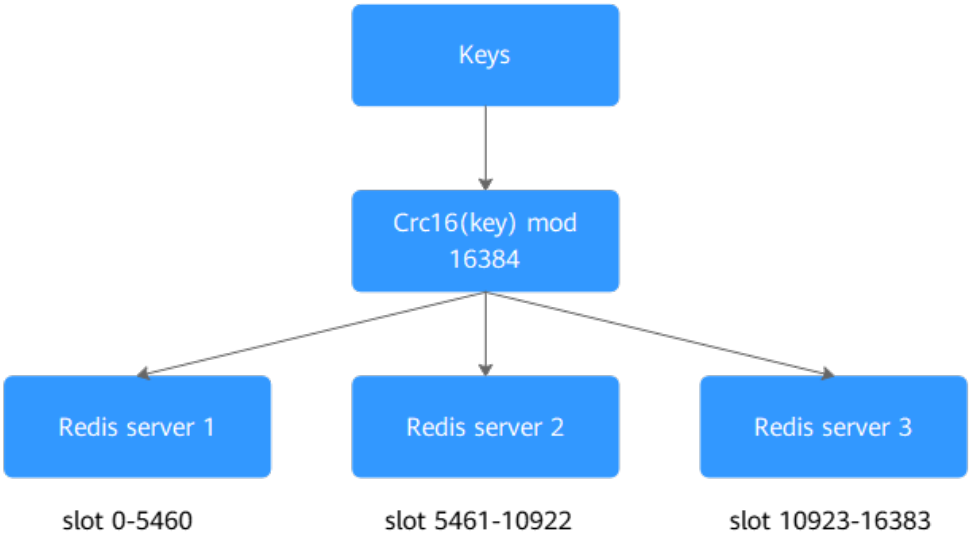


- Presharding

  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 4-5** 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 **Access in Different Languages**.

# 4.5 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.

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.

2. 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.

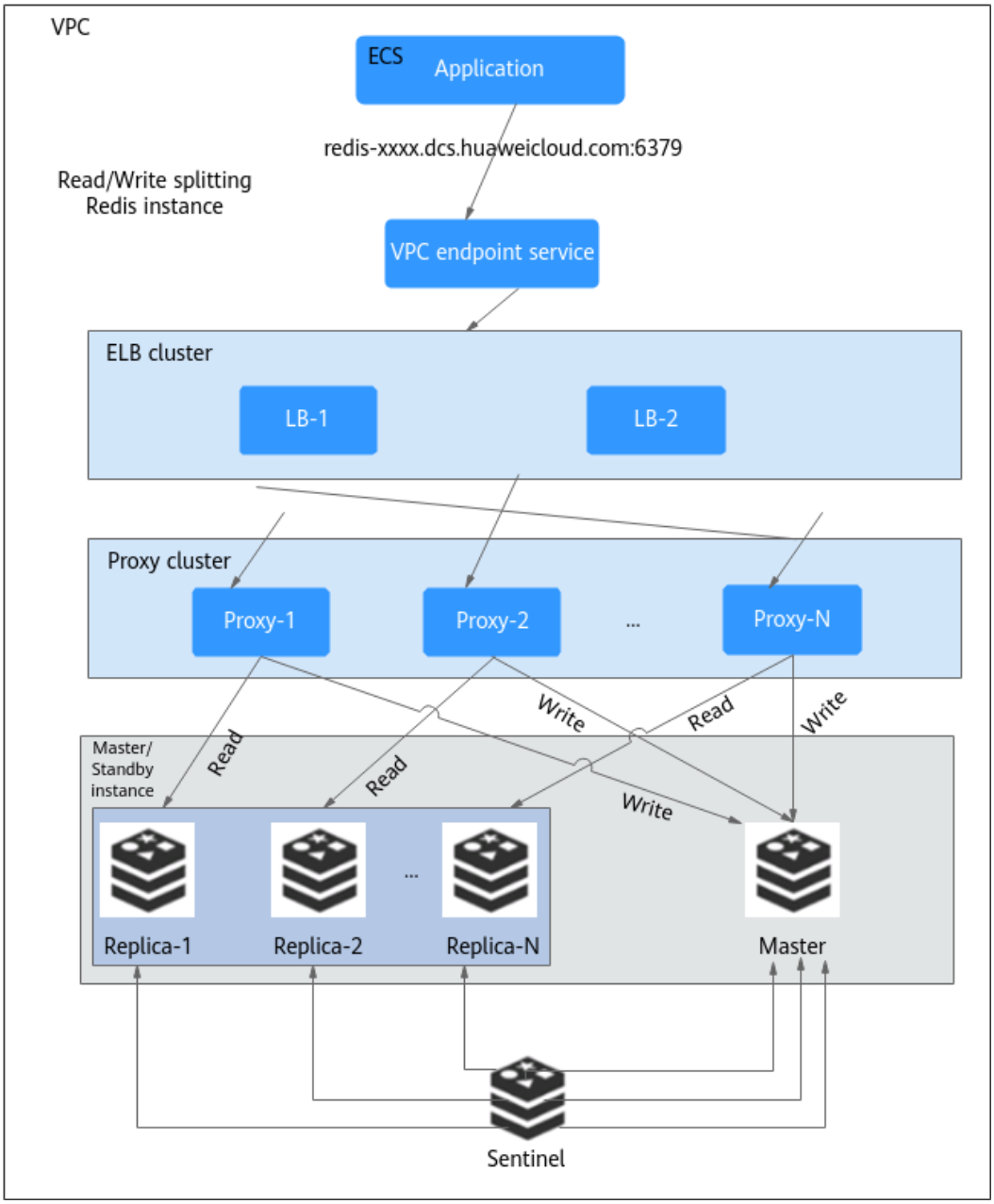
If your services are write-heavy, use master/standby or cluster instances.

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

**Figure 4-6** 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.

# 4.6 Comparing DCS Redis Instance Types

Table 4-3 describes the differences between different Redis instance types in terms of features and commands.

**Table 4-3** Differences between DCS instance types

| Item | Single-Node, Read/Write Splitting, or Master/ Standby | Proxy Cluster | Redis Cluster |
| --- | --- | --- | --- |
| Redis version compatibility | Redis 4.0/5.0/6.0 You can select a version when creating an instance. | Redis 4.0/5.0/6.0 You can select a version when creating an instance. | Redis 4.0/5.0/6.0 You can select a version when creating an instance. |

| Item | Single-Node, Read/Write Splitting, or Master/ Standby | Proxy Cluster | Redis Cluster |
| --- | --- | --- | --- |
| Support | • Keyspace notifications<br>• Pipelining | • Pipelining, **MSET** command, and **MGET** command<br>• **SCAN** command, **KEYS** command, and Redis Slow Log<br>• Pub/Sub | • Keyspace notifications<br>• **BRPOP**, **BLPOP**, and **BRPOPLPUSH** commands<br>• 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.<br>• 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**.<br>• Keyspace notifications are not supported. | • LUA script is restricted: All keys must be in the same hash slot. **Hash tags** are recommended.<br>• The client SDK must support Redis Cluster and be able to process MOVED errors.<br>• 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.<br>• When using keyspace notifications, establish connections with every Redis server and process events on each connection.<br>• 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 |
| --- | --- | --- | --- |
| Disabled commands | **Command Compatibility** lists disabled commands.<br>**Command Restrictions** lists the command restricted for read/write splitting instances. | **Command Compatibility** lists disabled commands.<br>**Command Restrictions** lists the command restricted for Proxy Cluster instances. | **Command Compatibility** lists disabled commands.<br>**Command Restrictions** lists the command restricted for Redis Cluster instances. |

| Item | Single-Node, Read/Write Splitting, or Master/ Standby | Proxy Cluster | Redis Cluster |
| --- | --- | --- | --- |
| Replicas | A single-node instance has only one replica.<br>By default, a master/standby or read/write splitting instance has two replicas, with one of them being the master.<br>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 master/ standby DCS 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?**

# 5 DCS Instance Specifications

## 5.1 Redis 4.0 and 5.0 Instance Specifications

This section describes DCS Redis 4.0 and 5.0 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. After an instance is created, this metric can be changed 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.
- 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 and 5.0 instances are available in single-node, master/standby, Proxy Cluster, Redis Cluster, and read/write splitting types.
- DCS Redis 4.0 and 5.0 instances support x86 and Arm architectures. x86 is recommended. Arm is unavailable in some regions.

### Single-Node Instances

Single-node DCS Redis 4.0 or 5.0 instances support x86 and Arm CPU architectures. The following table lists the specifications.

**Table 5-1** Specifications of single-node DCS Redis 4.0 or 5.0 instances

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 0.125 | 0.125 | 10,000/10,000 | 40/40 | 80,000 | x86: redis.single.xu1.tiny.128<br>Arm: redis.single.au1.tiny.128 |
| 0.25 | 0.25 | 10,000/10,000 | 80/80 | 80,000 | x86: redis.single.xu1.tiny.256<br>Arm: redis.single.au1.tiny.256 |
| 0.5 | 0.5 | 10,000/10,000 | 80/80 | 80,000 | x86: redis.single.xu1.tiny.512<br>Arm: redis.single.au1.tiny.512 |
| 1 | 1 | 10,000/50,000 | 80/80 | 80,000 | x86: redis.single.xu1.large.1<br>Arm: redis.single.au1.large.1 |
| 2 | 2 | 10,000/50,000 | 128/128 | 80,000 | x86: redis.single.xu1.large.2<br>Arm: redis.single.au1.large.2 |
| 4 | 4 | 10,000/50,000 | 192/192 | 80,000 | x86: redis.single.xu1.large.4<br>Arm: redis.single.au1.large.4 |

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 8 | 8 | 10,000/50,000 | 192/192 | 100,000 | x86: redis.single.xu1.large.8<br>Arm: redis.single.au1.large.8 |
| 16 | 16 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.single.xu1.large.16<br>Arm: redis.single.au1.large.16 |
| 24 | 24 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.single.xu1.large.24<br>Arm: redis.single.au1.large.24 |
| 32 | 32 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.single.xu1.large.32<br>Arm: redis.single.au1.large.32 |
| 48 | 48 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.single.xu1.large.48<br>Arm: redis.single.au1.large.48 |
| 64 | 64 | 10,000/50,000 | 384/384 | 100,000 | x86: redis.single.xu1.large.64<br>Arm: redis.single.au1.large.64 |

## Master/Standby Instances

Master/standby instances support x86 and Arm CPU architectures. An instance can have 2 to 5 replicas. For example, the specifications of an Arm-based instance can

be **Arm | master/standby | 2 replicas** or **Arm | master/standby | 5 replicas**. By default, a master/standby instance has one master node and two replicas (including the master replica).

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 5-2** only lists the specification names of x86- and Arm-based instances. The specification names reflect the number of replicas, for example, redis.ha.au1.large.**r2**.8 (master/standby | Arm | 2 replicas | 8 GB) and redis.ha.au1.large.**r3**.8 (master/standby | Arm | 3 replicas | 8 GB).
- IP addresses: Number of occupied IP addresses = Number of master nodes x Number of replicas. For example:

  2 replicas: Number of occupied IP addresses = 1 x 2 = 2

  3 replicas: Number of occupied IP addresses = 1 x 3 = 3

**Table 5-2** Specifications of master/standby DCS Redis 4.0 or 5.0 instances

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 0.125 | 0.125 | 10,000/10,000 | 40/40 | 80,000 | x86: redis.ha.xu1.tiny.r2.128<br>Arm: redis.ha.au1.tiny.r2.128 |
| 0.25 | 0.25 | 10,000/10,000 | 80/80 | 80,000 | x86: redis.ha.xu1.tiny.r2.256<br>Arm: redis.ha.au1.tiny.r2.256 |
| 0.5 | 0.5 | 10,000/10,000 | 80/80 | 80,000 | x86: redis.ha.xu1.tiny.r2.512<br>Arm: redis.ha.au1.tiny.r2.512 |

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 1 | 1 | 10,000/50,000 | 80/80 | 80,000 | x86: redis.ha.xu1.large.r2.1<br>Arm: redis.ha.au1.large.r2.1 |
| 2 | 2 | 10,000/50,000 | 128/128 | 80,000 | x86: redis.ha.xu1.large.r2.2<br>Arm: redis.ha.au1.large.r2.2 |
| 4 | 4 | 10,000/50,000 | 192/192 | 80,000 | x86: redis.ha.xu1.large.r2.4<br>Arm: redis.ha.au1.large.r2.4 |
| 8 | 8 | 10,000/50,000 | 192/192 | 100,000 | x86: redis.ha.xu1.large.r2.8<br>Arm: redis.ha.au1.large.r2.8 |
| 16 | 16 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.ha.xu1.large.r2.16<br>Arm: redis.ha.au1.large.r2.16 |
| 24 | 24 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.ha.xu1.large.r2.24<br>Arm: redis.ha.au1.large.r2.24 |

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 32 | 32 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.ha.xu1.large.r2.32<br>Arm: redis.ha.au1.large.r2.32 |
| 48 | 48 | 10,000/50,000 | 256/256 | 100,000 | x86: redis.ha.xu1.large.r2.48<br>Arm: redis.ha.au1.large.r2.48 |
| 64 | 64 | 10,000/50,000 | 384/384 | 100,000 | x86: redis.ha.xu1.large.r2.64<br>Arm: redis.ha.au1.large.r2.64 |

## Proxy Cluster Instances

Proxy Cluster instances support x86 and Arm CPU architectures. **Table 5-3** lists the specifications.

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 4-1**. When buying an instance, you can customize the size of a single shard.

NOTE

- 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 5-3** Specifications of Proxy Cluster DCS Redis 4.0 and 5.0 instances

| Specification (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 4 | 4 | 3 | 20,000/20,000 | 2304/2304 | 240,000 | x86: redis.proxy.xu1.large.4<br>Arm: redis.proxy.au1.large.4 |
| 8 | 8 | 3 | 30,000/30,000 | 2304/2304 | 240,000 | x86: redis.proxy.xu1.large.8<br>Arm: redis.proxy.au1.large.8 |
| 16 | 16 | 3 | 30,000/30,000 | 2304/2304 | 240,000 | x86: redis.proxy.xu1.large.16<br>Arm: redis.proxy.au1.large.16 |

| Specification (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 24 | 24 | 3 | 30,000/30,000 | 2304/2304 | 240,000 | x86: redis.proxy.xu1.large.24<br>Arm: redis.proxy.au1.large.24 |
| 32 | 32 | 3 | 30,000/30,000 | 2304/2304 | 240,000 | x86: redis.proxy.xu1.large.32<br>Arm: redis.proxy.au1.large.32 |
| 48 | 48 | 6 | 60,000/60,000 | 4608/4608 | 480,000 | x86: redis.proxy.xu1.large.48<br>Arm: redis.proxy.au1.large.48 |
| 64 | 64 | 8 | 80,000/80,000 | 6144/6144 | 640,000 | x86: redis.proxy.xu1.large.64<br>Arm: redis.proxy.au1.large.64 |
| 96 | 96 | 12 | 120,000/120,000 | 9216/9216 | 960,000 | x86: redis.proxy.xu1.large.96<br>Arm: redis.proxy.au1.large.96 |
| 128 | 128 | 16 | 160,000/160,000 | 10,000/10,000 | 1,280,000 | x86: redis.proxy.xu1.large.128<br>Arm: redis.proxy.au1.large.128 |

| Specification (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 192 | 192 | 24 | 200,000/240,000 | 10,000/10,000 | 1,920,000 | x86: redis.proxy.xu1.large.192<br>Arm: redis.proxy.au1.large.192 |
| 256 | 256 | 32 | 200,000/320,000 | 10,000/10,000 | > 2,000,000 | x86: redis.proxy.xu1.large.256<br>Arm: redis.proxy.au1.large.256 |
| 384 | 384 | 48 | 200,000/480,000 | 10,000/10,000 | > 2,000,000 | x86: redis.proxy.xu1.large.384<br>Arm: redis.proxy.au1.large.384 |
| 512 | 512 | 64 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | x86: redis.proxy.xu1.large.512<br>Arm: redis.proxy.au1.large.512 |
| 768 | 768 | 96 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | x86: redis.proxy.xu1.large.768<br>Arm: redis.proxy.au1.large.768 |
| 1024 | 1024 | 128 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | x86: redis.proxy.xu1.large.1024<br>Arm: redis.proxy.au1.large.1024 |

## Redis Cluster Instances

Redis Cluster instances support x86 and Arm CPU architectures. Each instance 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 multiple replicas are as follows:

- The available memory, shard quantity (master node quantity), maximum number of connections, assured/maximum bandwidth, and QPS are the same.
- Specification name: **Table 5-4** 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.au1.large.**r2**.24 (Redis Cluster | Arm | 2 replicas | 24 GB) and redis.cluster.au1.large.**r3**.24 (Redis Cluster | Arm | 3 replicas | 24 GB).
- IP addresses: Number of occupied IP addresses = Number of shards x 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 5-4** Specifications of Redis Cluster DCS Redis 4.0 and 5.0 instances

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/ Limit) (Count) | Assured/ Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 4 | 4 | 3 | 30,000 / 150,000 | 2304/2304 | 240,000 | x86: redis.cluster.xu1.large.r2.4<br>Arm: redis.cluster.au1.large.r2.4 |
| 8 | 8 | 3 | 30,000 / 150,000 | 2304/2304 | 240,000 | x86: redis.cluster.xu1.large.r2.8<br>Arm: redis.cluster.au1.large.r2.8 |
| 16 | 16 | 3 | 30,000 / 150,000 | 2304/2304 | 240,000 | x86: redis.cluster.xu1.large.r2.16<br>Arm: redis.cluster.au1.large.r2.16 |
| 24 | 24 | 3 | 30,000 / 150,000 | 2304/2304 | 300,000 | x86: redis.cluster.xu1.large.r2.24<br>Arm: redis.cluster.au1.large.r2.24 |
| 32 | 32 | 3 | 30,000 / 150,000 | 2304/2304 | 300,000 | x86: redis.cluster.xu1.large.r2.32<br>Arm: redis.cluster.au1.large.r2.32 |
| 48 | 48 | 6 | 60,000 / 300,000 | 4608/4608 | > 300,000 | x86: redis.cluster.xu1.large.r2.48<br>Arm: redis.cluster.au1.large.r2.48 |

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/ Limit) (Count) | Assured/ Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 64 | 64 | 8 | 80,000 / 400,000 | 6144/6144 | 500,000 | x86: redis.cluster.xu1.large.r2.64<br>Arm: redis.cluster.au1.large.r2.64 |
| 96 | 96 | 12 | 120,000 / 600,000 | 9216/9216 | > 500,000 | x86: redis.cluster.xu1.large.r2.96<br>Arm: redis.cluster.au1.large.r2.96 |
| 128 | 128 | 16 | 160,000 / 800,000 | 12,288/12,288 | 1,000,000 | x86: redis.cluster.xu1.large.r2.128<br>Arm: redis.cluster.au1.large.r2.128 |
| 192 | 192 | 24 | 240,000 / 1,200,000 | 18,432/18,432 | > 1,000,000 | x86: redis.cluster.xu1.large.r2.192<br>Arm: redis.cluster.au1.large.r2.192 |
| 256 | 256 | 32 | 320,000 / 1,600,000 | 24,576/24,576 | > 2,000,000 | x86: redis.cluster.xu1.large.r2.256<br>Arm: redis.cluster.au1.large.r2.256 |
| 384 | 384 | 48 | 480,000 / 2,400,000 | 36,864/36,864 | > 2,000,000 | x86: redis.cluster.xu1.large.r2.384<br>Arm: redis.cluster.au1.large.r2.384 |

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/ Limit) (Count) | Assured/ Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 512 | 512 | 64 | 640,000 / 3,200,000 | 49,152/49,152 | > 2,000,000 | x86: redis.cluster.xu1.large.r2.512<br>Arm: redis.cluster.au1.large.r2.512 |
| 768 | 768 | 96 | 960,000 / 4,800,000 | 73,728/73,728 | > 2,000,000 | x86: redis.cluster.xu1.large.r2.768<br>Arm: redis.cluster.au1.large.r2.768 |
| 1024 | 1024 | 128 | 1,280,000 / 6,400,000 | 98,304/98,304 | > 2,000,000 | x86: redis.cluster.xu1.large.r2.1024<br>Arm: redis.cluster.au1.large.r2.1024 |
| 2048 | 2048 | 128 | 1,280,000 / 6,400,000 | 98,304/98,304 | >2,000,000 | x86: redis.cluster.xu1.large.r2.2048<br>Arm: redis.cluster.au1.large.r2.2048 |

## Read/Write Splitting Instances

- Currently, read/write splitting instances only support the x86 CPU architecture. **Table 5-5** lists the specifications.
- 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 per node (QPS) = Reference performance (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 5-5** Specifications of read/write splitting DCS Redis 4.0 or 5.0 instances

| Specification | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.1 |
| 1 | 1 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.1 |
| 1 | 1 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.1 |
| 1 | 1 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.1 |
| 1 | 1 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.1 |

| Specification | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.2 |
| 2 | 2 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.2 |
| 2 | 2 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.2 |
| 2 | 2 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.2 |
| 2 | 2 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.2 |
| 4 | 4 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.4 |
| 4 | 4 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.4 |
| 4 | 4 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.4 |
| 4 | 4 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.4 |

| Specification | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 4 | 4 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.4 |
| 8 | 8 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.8 |
| 8 | 8 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.8 |
| 8 | 8 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.8 |
| 8 | 8 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.8 |
| 8 | 8 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.8 |
| 16 | 16 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.16 |
| 16 | 16 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.16 |
| 16 | 16 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.16 |

| Specification | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 16 | 16 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.16 |
| 16 | 16 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.16 |
| 32 | 32 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.32 |
| 32 | 32 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.32 |
| 32 | 32 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.32 |
| 32 | 32 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.32 |
| 32 | 32 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.32 |
| 64 | 64 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.64 |
| 64 | 64 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.64 |

| Specification | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 64 | 64 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.64 |
| 64 | 64 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.64 |
| 64 | 64 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.64 |

# 5.2 Redis 6.0 Instance Specifications

This section describes DCS Redis 6.0 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.
- QPS represents queries per second, which is the number of commands processed per second.
- 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%?**

## Master/Standby

**Table 5-6** Specifications of master/standby DCS Redis 6.0 basic edition instances

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 1 | 1 | 10,000/50,000 | 80/80 | 80,000 | redis.ha.xu1.large.r2.1 |
| 2 | 2 | 10,000/50,000 | 128/128 | 80,000 | redis.ha.xu1.large.r2.2 |
| 4 | 4 | 10,000/50,000 | 192/192 | 80,000 | redis.ha.xu1.large.r2.4 |
| 8 | 8 | 10,000/50,000 | 192/192 | 100,000 | redis.ha.xu1.large.r2.8 |
| 16 | 16 | 10,000/50,000 | 256/256 | 100,000 | redis.ha.xu1.large.r2.16 |
| 24 | 24 | 10,000/50,000 | 256/256 | 100,000 | redis.ha.xu1.large.r2.24 |
| 32 | 32 | 10,000/50,000 | 256/256 | 100,000 | redis.ha.xu1.large.r2.32 |
| 48 | 48 | 10,000/50,000 | 256/256 | 100,000 | redis.ha.xu1.large.r2.48 |
| 64 | 64 | 10,000/50,000 | 384/384 | 100,000 | redis.ha.xu1.large.r2.64 |

## Single-Node

**Table 5-7** Specifications of single-node DCS Redis 6.0 basic edition instances

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 0.125 | 0.125 | 10,000/10,000 | 40/40 | 80,000 | redis.single.xu1.tiny.128 |
| 0.25 | 0.25 | 10,000/10,000 | 80/80 | 80,000 | redis.single.xu1.tiny.256 |

| Total Memory (GB) | Available Memory (GB) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|
| 0.5 | 0.5 | 10,000/10,000 | 80/80 | 80,000 | redis.single.xu1.tiny.512 |
| 1 | 1 | 10,000/50,000 | 80/80 | 80,000 | redis.single.xu1.large.1 |
| 2 | 2 | 10,000/50,000 | 128/128 | 80,000 | redis.single.xu1.large.2 |
| 4 | 4 | 10,000/50,000 | 192/192 | 80,000 | redis.single.xu1.large.4 |
| 8 | 8 | 10,000/50,000 | 192/192 | 100,000 | redis.single.xu1.large.8 |
| 16 | 16 | 10,000/50,000 | 256/256 | 100,000 | redis.single.xu1.large.16 |
| 24 | 24 | 10,000/50,000 | 256/256 | 100,000 | redis.single.xu1.large.24 |
| 32 | 32 | 10,000/50,000 | 256/256 | 100,000 | redis.single.xu1.large.32 |
| 48 | 48 | 10,000/50,000 | 256/256 | 100,000 | redis.single.xu1.large.48 |
| 64 | 64 | 10,000/50,000 | 384/384 | 100,000 | redis.single.xu1.large.64 |

## Proxy Cluster

**Table 5-8** lists the specifications supported by 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 4-1**. When buying an instance, you can customize the size of a single shard.

NOTE

- 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 5-8** Specifications of Proxy Cluster DCS Redis 6.0 basic edition instances

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 4 | 4 | 3 | 20,000/20,000 | 2,304/2,304 | 240,000 | redis.proxy.xu1.large.4 |
| 8 | 8 | 3 | 30,000/30,000 | 2,304/2,304 | 240,000 | redis.proxy.xu1.large.8 |
| 16 | 16 | 3 | 30,000/30,000 | 2,304/2,304 | 240,000 | redis.proxy.xu1.large.16 |
| 24 | 24 | 3 | 30,000/30,000 | 2,304/2,304 | 240,000 | redis.proxy.xu1.large.24 |
| 32 | 32 | 3 | 30,000/30,000 | 2,304/2,304 | 240,000 | redis.proxy.xu1.large.32 |
| 48 | 48 | 6 | 60,000/60,000 | 4,608/4,608 | 480,000 | redis.proxy.xu1.large.48 |
| 64 | 64 | 8 | 80,000/80,000 | 6,144/6,144 | 640,000 | redis.proxy.xu1.large.64 |
| 96 | 96 | 12 | 120,000/120,000 | 9,216/9,216 | 960,000 | redis.proxy.xu1.large.96 |
| 128 | 128 | 16 | 160,000/160,000 | 10,000/10,000 | 1,280,000 | redis.proxy.xu1.large.128 |

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 192 | 192 | 24 | 200,000/240,000 | 10,000/10,000 | 1,920,000 | redis.proxy.xu1.large.192 |
| 256 | 256 | 32 | 200,000/320,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.256 |
| 384 | 384 | 48 | 200,000/480,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.384 |
| 512 | 512 | 64 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.512 |
| 768 | 768 | 96 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.768 |
| 1024 | 1024 | 128 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.1024 |
| 2048 | 2048 | 128 | 200,000/500,000 | 10,000/10,000 | > 2,000,000 | redis.proxy.xu1.large.2048 |

## Redis Cluster

- Specification code: **Table 5-9** only lists the specification codes of instances with 2 replicas (default). The specification names reflect the number of replicas, for example, redis.cluster.xu1.large.r**2**.4 (2 replicas | 4 GB) and redis.cluster.xu1.large.r**1**.4 (1 replica | 4 GB).
- IP addresses: Number of occupied IP addresses = Number of shards x Number of replicas. For example:

  24 GB | Redis Cluster | 2 replicas: Number of occupied IP addresses = 3 x 2 = 6
- Available memory per node = Instance available memory/Master node quantity. For example:

  For example, a 24 GB 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 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 5-9** Specifications of Redis Cluster DCS Redis 6.0 instances

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 4 | 4 | 3 | 30,000/150,000 | 2304/2304 | 240,000 | redis.cluster.xu1.large.r2.4 |
| 8 | 8 | 3 | 30,000/150,000 | 2304/2304 | 240,000 | redis.cluster.xu1.large.r2.8 |
| 16 | 16 | 3 | 30,000/150,000 | 2304/2304 | 240,000 | redis.cluster.xu1.large.r2.16 |
| 24 | 24 | 3 | 30,000/150,000 | 2304/2304 | 300,000 | redis.cluster.xu1.large.r2.24 |
| 32 | 32 | 3 | 30,000/150,000 | 2304/2304 | 300,000 | redis.cluster.xu1.large.r2.32 |
| 48 | 48 | 6 | 60,000/300,000 | 4608/4608 | > 300,000 | redis.cluster.xu1.large.r2.48 |
| 64 | 64 | 8 | 80,000/400,000 | 6144/6144 | 500,000 | redis.cluster.xu1.large.r2.64 |
| 96 | 96 | 12 | 120,000/600,000 | 9216/9216 | > 500,000 | redis.cluster.xu1.large.r2.96 |
| 128 | 128 | 16 | 160,000/800,000 | 12,288/12,288 | 1,000,000 | redis.cluster.xu1.large.r2.128 |

| Total Memory (GB) | Available Memory (GB) | Shards (Master Nodes) | Max. Connections (Default/Limit) (Count) | Assured/Maximum Bandwidth (Mbit/s) | Reference Performance (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|
| 192 | 192 | 24 | 240,000/1,200,000 | 18,432/18,432 | > 1,000,000 | redis.cluster.xu1.large.r2.192 |
| 256 | 256 | 32 | 320,000/1,600,000 | 24,576/24,576 | > 2,000,000 | redis.cluster.xu1.large.r2.256 |
| 384 | 384 | 48 | 480,000/2,400,000 | 36,864/36,864 | > 2,000,000 | redis.cluster.xu1.large.r2.384 |
| 512 | 512 | 64 | 640,000/3,200,000 | 49,152/49,152 | > 2,000,000 | redis.cluster.xu1.large.r2.512 |
| 768 | 768 | 96 | 960,000/4,800,000 | 73,728/73,728 | > 2,000,000 | redis.cluster.xu1.large.r2.768 |
| 1024 | 1024 | 128 | 1,280,000/6,400,000 | 98,304/98,304 | > 2,000,000 | redis.cluster.xu1.large.r2.1024 |
| 2048 | 2048 | 128 | 2,560,000/12,800,000 | 98,304/98,304 | > 2,000,000 | redis.cluster.xu1.large.r2.2048 |

## Read/Write Splitting

- For details about the specifications of read/write splitting instances, see **Table 5-10**.
- 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 per node (QPS) = Reference performance (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 5-10** Read/Write splitting DCS Redis 6.0 instance specifications

| Total Memory | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.1 |
| 1 | 1 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.1 |
| 1 | 1 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.1 |
| 1 | 1 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.1 |
| 1 | 1 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.1 |
| 2 | 2 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.2 |
| 2 | 2 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.2 |

| Total Memory | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.2 |
| 2 | 2 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.2 |
| 2 | 2 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.2 |
| 4 | 4 | 2 | 20,000 | 96 | 48 | 160,000 | 80,000 | redis.ha.xu1.large.p2.4 |
| 4 | 4 | 3 | 30,000 | 144 | 48 | 240,000 | 80,000 | redis.ha.xu1.large.p3.4 |
| 4 | 4 | 4 | 40,000 | 192 | 48 | 320,000 | 80,000 | redis.ha.xu1.large.p4.4 |
| 4 | 4 | 5 | 50,000 | 240 | 48 | 400,000 | 80,000 | redis.ha.xu1.large.p5.4 |
| 4 | 4 | 6 | 60,000 | 288 | 48 | 480,000 | 80,000 | redis.ha.xu1.large.p6.4 |
| 8 | 8 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.8 |

| Total Memory | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 8 | 8 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.8 |
| 8 | 8 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.8 |
| 8 | 8 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.8 |
| 8 | 8 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.8 |
| 16 | 16 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.16 |
| 16 | 16 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.16 |
| 16 | 16 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.16 |
| 16 | 16 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.16 |
| 16 | 16 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.16 |

| Total Memory | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 32 | 32 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.32 |
| 32 | 32 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.32 |
| 32 | 32 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.32 |
| 32 | 32 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.32 |
| 32 | 32 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.32 |
| 64 | 64 | 2 | 20,000 | 192 | 96 | 160,000 | 80,000 | redis.ha.xu1.large.p2.64 |
| 64 | 64 | 3 | 30,000 | 288 | 96 | 240,000 | 80,000 | redis.ha.xu1.large.p3.64 |
| 64 | 64 | 4 | 40,000 | 384 | 96 | 320,000 | 80,000 | redis.ha.xu1.large.p4.64 |
| 64 | 64 | 5 | 50,000 | 480 | 96 | 400,000 | 80,000 | redis.ha.xu1.large.p5.64 |

| Total Memory | Available Memory (GB) | Replicas (Including Masters) | Max. Connections (Default/ Limit) | Bandwidth Limit (MB/s) | Bandwidth Limit per Redis Server (MB/s) | Reference Performance (QPS) | Reference Performance per Node (QPS) | Specification Code (spec_code in the API) |
|---|---|---|---|---|---|---|---|---|
| 64 | 64 | 6 | 60,000 | 576 | 96 | 480,000 | 80,000 | redis.ha.xu1.large.p6.64 |

# 6 Command Compatibility

## 6.1 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 6-1** and **Table 6-2** list commands supported by single-node, master/standby, and Redis Cluster DCS Redis 4.0 instances.
- **Table 6-3** and **Table 6-4** list the Redis commands supported by Proxy Cluster DCS Redis 4.0 instances.
- **Table 6-5** and **Table 6-6** 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 6-1** 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 | APPEND | HDEL | BLPOP | SADD | ZADD | FLUSHALL |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | CLIENT KILL |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | CLIENT LIST |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | CLIENT GETNAME |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | CLIENT SETNAME |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | CONFIG GET |
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | MONITOR |
| SORT | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | SLOWLOG |
| TTL | MGET | HSCAN | RPOP | SUNION | ZSCORE | ROLE |
| TYPE | MSET | HSTRLEN | RPOPLPU | SUNIONSTORE | ZUNIONSTORE | SWAPDB |

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

**Table 6-2** Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 4.0 instances (2)

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo |
|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST |
| - | SUBSCRIBE | WATCH | SELECT (not supported by Redis Cluster instances) | SCRIPT KILL | GEORADIUS |
| - | UNSUBSCRIBE | - | - | SCRIPT LOAD | GEORADIUSBYMEMBER |

**Table 6-3** 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 | FLUSHALL (FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | ROLE |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | MEMORY |
| RENAME | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | COMMAND |
| RENAMENX | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | COMMAND COUNT |
| RESTORE | INCR | HSET | LREM | SPOP | ZREVRANGE | COMMAND GETKEYS |
| SORT | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND INFO |
| TTL | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | CONFIG GET |
| TYPE | MGET | HSCAN | RPOP | SUNION | ZSCORE | CONFIG RESETSTAT |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| SCAN | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | CONFIG REWRITE |
| OBJECT | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG SET |
| PEXPIRE | PSETEX | HKEYS | RPUSHX | - | ZSCAN | - |
| PEXPIREAT | SET | - | LPUSH | - | ZRANGEBYLEX | - |
| EXPIREAT | SETBIT | - | - | - | ZLEXCOUNT | - |
| KEYS | SETEX | - | - | - | ZREMRANGEBYSCORE | - |
| TOUCH | SETNX | - | - | - | ZREM | - |
| UNLINK | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |
| RANDOMKEY | STRLEN | - | - | - | ZREVRANGEBYLEX | - |
| - | BITFIELD | - | - | - | - | - |
| - | GETBIT | - | - | - | - | - |

**Table 6-4** Commands supported by Proxy Cluster DCS Redis 4.0 instances (2)

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo | Cluster |
|---|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD | CLUSTER INFO |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH | CLUSTER NODES |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS | CLUSTER SLOTS |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST | CLUSTER ADDSLOTS |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS | ASKING |

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo | Cluster |
|---|---|---|---|---|---|---|
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER | READONLY |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH | READWRITE |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE | - |
| - | - | - | CLIENT SETNAME | - | - | - |

NOTE

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

**Table 6-5** 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 | FLUSHALL<br>(FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | MONITOR |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | SLOWLOG |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | ROLE |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | SWAPDB |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | MEMORY |
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND |
| SORT | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | COMMAND COUNT |
| TTL | MGET | HSCAN | RPOP | SUNION | ZSCORE | COMMAND GETKEYS |
| TYPE | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | COMMAND INFO |
| SCAN | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG GET |
| OBJECT | PSETEX | - | RPUSHX | - | ZSCAN | CONFIG RESETSTAT |
| PEXPIRE | SET | - | LPUSH | - | ZRANGEBYLEX | CONFIG REWRITE |
| PEXPIREAT | SETBIT | - | - | - | ZLEXCOUNT | CONFIG SET |
| EXPIREAT | SETEX | - | - | - | ZREMRANGEBYSCORE | - |
| KEYS | SETNX | - | - | - | ZREM | - |
| TOUCH | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| UNLINK | STRLEN | - | - | - | ZREVRANGEBYLEX | - |
| - | BITFIELD | - | - | - | - | - |
| - | GETBIT | - | - | - | - | - |

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

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo |
|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS |
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE |
| - | - | - | CLIENT SETNAME | - | - |

## Commands Disabled by DCS for Redis 4.0

The following lists commands disabled by DCS for Redis 4.0.

**Table 6-7** 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 6-8** Redis commands disabled in Proxy Cluster DCS Redis 4.0 instances

| Generic (Key) | Server | Sorted Set |
| --- | --- | --- |
| MIGRATE | BGREWRITEAOF | BZPOPMAX |
| 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 6-9** 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 6-10** 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 6-11** Commands that can be renamed

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

## 6.2 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 6-12** and **Table 6-13** list commands supported by single-node, master/standby, and Redis Cluster DCS Redis 5.0 instances.
- **Table 6-14** and **Table 6-15** list commands supported by Proxy Cluster DCS for Redis 5.0 instances.

- **Table 6-16** and **Table 6-17** 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 6-12** 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 | APPEND | HDEL | BLPOP | SADD | ZADD | FLUSHALL |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | KEYS |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | CLIENT KILL |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | CLIENT LIST |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | CLIENT GETNAME |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | CLIENT SETNAME |
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | CONFIG GET |

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

**Table 6-13** Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 5.0 instances (2)

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

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo | Stream |
|---|---|---|---|---|---|---|
| - | SUBSCRIBE | WATCH | SELECT (not supported by Redis Cluster instances) | SCRIPT KILL | GEORADIUS | XGROUP |
| - | UNSUBSCRIBE | - | - | SCRIPT LOAD | GEORADIUSBYMEMBER | XINFO |
| - | - | - | - | - | - | XLEN |
| - | - | - | - | - | - | XPENDING |
| - | - | - | - | - | - | XRANGE |
| - | - | - | - | - | - | XREAD |
| - | - | - | - | - | - | XREADGROUP |
| - | - | - | - | - | - | XREVRANGE |
| - | - | - | - | - | - | XTRIM |

**Table 6-14** 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 | FLUSHALL<br>(FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | ROLE |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | MEMORY |
| RENAME | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | COMMAND |
| RENAMENX | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | COMMAND COUNT |
| RESTORE | INCR | HSET | LREM | SPOP | ZREVRANGE | COMMAND GETKEYS |
| SORT | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND INFO |
| TTL | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | CONFIG GET |
| TYPE | MGET | HSCAN | RPOP | SUNION | ZSCORE | CONFIG RESETSTAT |
| SCAN | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | CONFIG REWRITE |
| OBJECT | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG SET |
| PEXPIRE | PSETEX | HKEYS | RPUSHX | - | ZSCAN | - |
| PEXPIREAT | SET | - | LPUSH | - | ZRANGEBYLEX | - |
| EXPIREAT | SETBIT | - | - | - | ZLEXCOUNT | - |
| KEYS | SETEX | - | - | - | ZREMRANGEBYSCORE | - |
| UNLINK | SETNX | - | - | - | ZREM | - |
| TOUCH | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| RANDOMKEY | STRLEN | - | - | - | ZPOPMAX | - |
| - | BITFIELD | - | - | - | ZPOPMIN | - |
| - | GETBIT | - | - | - | BZPOPMAX | - |
| - | - | - | - | - | BZPOPMIN | - |
| - | - | - | - | - | ZREVRANGEBYLEX | - |

**Table 6-15** Commands supported by Proxy Cluster DCS Redis 5.0 instances (2)

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo | Cluster |
|---|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD | CLUSTER INFO |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH | CLUSTER NODES |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS | CLUSTER SLOTS |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST | CLUSTER ADDSLOTS |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS | ASKING |
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER | READONLY |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH | READWRITE |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE | - |

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | 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 6-16** 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 | FLUSHALL<br>(FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | MONITOR |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | SLOWLOG |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | ROLE |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | SWAPDB |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | MEMORY |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND |
| SORT | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | COMMAND COUNT |
| TTL | MGET | HSCAN | RPOP | SUNION | ZSCORE | COMMAND GETKEYS |
| TYPE | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | COMMAND INFO |
| SCAN | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG GET |
| OBJECT | PSETEX | - | RPUSHX | - | ZSCAN | CONFIG RESETSTAT |
| PEXPIRE | SET | - | LPUSH | - | ZRANGEBYLEX | CONFIG REWRITE |
| PEXPIREAT | SETBIT | - | - | - | ZLEXCOUNT | CONFIG SET |
| EXPIREAT | SETEX | - | - | - | ZREMRANGEBYSCORE | - |
| KEYS | SETNX | - | - | - | ZREM | - |
| UNLINK | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |
| TOUCH | STRLEN | - | - | - | BZPOPMAX | - |
| - | BITFIELD | - | - | - | BZPOPMIN | - |
| - | GETBIT | - | - | - | ZPOPMAX | - |
| - | - | - | - | - | ZPOPMIN | - |
| - | - | - | - | - | ZREVRANGEBYLEX | - |

**Table 6-17** Commands supported by read/write splitting DCS Redis 5.0 instances (2)

| HyperLogLog | Pub/Sub | Transactions | Connection | Scripting | Geo | Stream |
|---|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD | XACK |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH | XADD |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS | XCLAIM |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST | XDEL |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS | XGROUP |
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER | XINFO |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH | XLEN |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE | XPENDING |
| - | - | - | CLIENT SETNAME | - | - | XRANGE |
| - | - | - | - | - | - | XREAD |
| - | - | - | - | - | - | XREADGROUP |
| - | - | - | - | - | - | XREVRANGE |
| - | - | - | - | - | - | XTRIM |

## Commands Disabled by DCS for Redis 5.0

The following lists commands disabled by DCS for Redis 5.0.

**Table 6-18** 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 6-19** 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 6-20** 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 6-21** 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 6-22** Commands that can be renamed

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

## 6.3 Commands Supported and Disabled by DCS for Redis 6.0

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

- **Table 6-23** and **Table 6-24** list commands supported by single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances.
- **Table 6-25** and **Table 6-26** list commands supported by Proxy Cluster DCS for Redis 6.0 instances.
- **Table 6-27** and **Table 6-28** 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 6-23** 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 | APPEND | HDEL | BLPOP | SADD | ZADD | FLUSHALL |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | CONFIG GET |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | MONITOR |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | SLOWLOG |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | ROLE |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | SWAPDB |
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | MEMORY |
| SORT | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | CONFIG |
| TTL | MGET | HSCAN | RPOP | SUNION | ZSCORE | ACL |
| TYPE | MSET | HSTRLEN | RPOPLPU | SUNIONSTORE | ZUNIONSTORE | COMMAND |
| SCAN | MSETNX | HLEN | RPOPLPUSH | SSCAN | ZINTERSTORE | - |
| OBJECT | PSETEX | - | RPUSH | SMISMEMBER | ZSCAN | - |
| PEXPIREAT | SET | - | RPUSHX | - | ZRANGEBYLEX | - |
| PEXPIRE | SETBIT | - | LPUSH | - | ZLEXCOUNT | - |

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

**Table 6-24** Commands supported by single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances (2)

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

| HyperLoglog | Pub/Sub | Transactions | Connection | Scripting | Geo | Stream |
|---|---|---|---|---|---|---|
| - | SUBSCRIBE | WATCH | SELECT (not supported by Redis Cluster instances) | SCRIPT KILL | GEORADIUS | XGROUP |
| - | UNSUBSCRIBE | - | CLIENT CACHING | SCRIPT LOAD | GEORADIUSBYMEMBER | XINFO |
| - | - | - | CLIENT GETREDIR | - | - | XLEN |
| - | - | - | CLIENT INFO | - | - | XPENDING |
| - | - | - | CLIENT TRACKING | - | - | XRANGE |
| - | - | - | CLIENT TRACKINGINFO | - | - | XREAD |
| - | - | - | CLIENT UNPAUSE | - | - | XREADGROUP |
| - | - | - | CLIENT KILL | - | - | XREVRANGE |
| - | - | - | CLIENT LIST | - | - | XTRIM |
| - | - | - | CLIENT GETNAME | - | - | XAUTOCLAIM |
| - | - | - | CLIENT SETNAME | - | - | XGROUP CREATECONSUMER |
| - | - | - | HELLO | - | - | - |
| - | - | - | RESET | - | - | - |

**Table 6-25** 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 | FLUSHALL (FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | ROLE |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | MEMORY |
| RENAME | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | COMMAND |
| RENAMENX | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | COMMAND COUNT |
| RESTORE | INCR | HSET | LREM | SPOP | ZREVRANGE | COMMAND GETKEYS |
| SORT | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND INFO |
| TTL | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | CONFIG GET |
| TYPE | MGET | HSCAN | RPOP | SUNION | ZSCORE | CONFIG RESETSTAT |
| SCAN | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | CONFIG REWRITE |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| OBJECT | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG SET |
| PEXPIRE | PSETEX | HKEYS | RPUSHX | SMISMEMBER | ZSCAN | - |
| PEXPIREAT | SET | HRANDFIELD | LPUSH | - | ZRANGEBYLEX | - |
| EXPIREAT | SETBIT | - | BLMOVE | - | ZLEXCOUNT | - |
| KEYS | SETEX | - | LMOVE | - | ZREMRANGEBYSCORE | - |
| UNLINK | SETNX | - | LPOS | - | ZREM | - |
| TOUCH | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |
| RANDOMKEY | STRLEN | - | - | - | ZPOPMAX | - |
| COPY | BITFIELD | - | - | - | ZPOPMIN | - |
| - | GETBIT | - | - | - | BZPOPMAX | - |
| - | BITFIELD_RO | - | - | - | BZPOPMIN | - |
| - | GETDEL | - | - | - | ZREVRANGEBYLEX | - |
| - | GETEX | - | - | - | ZDIFF | - |
| - | - | - | - | - | ZDIFFSTORE | - |
| - | - | - | - | - | ZINTER | - |
| - | - | - | - | - | ZMSCORE | - |
| - | - | - | - | - | ZRANDMEMBER | - |
| - | - | - | - | - | ZRANGESTORE | - |
| - | - | - | - | - | ZUNION | - |

**Table 6-26** Commands supported by Proxy Cluster DCS Redis 6.0 instances (2)

| HyperLoglog | Pub/Sub | Transactions | Connection | Scripting | Geo | Cluster |
|---|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD | CLUSTER INFO |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH | CLUSTER NODES |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS | CLUSTER SLOTS |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST | CLUSTER ADDSLOTS |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS | ASKING |
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER | READONLY |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH | READWRITE |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE | - |
| - | - | - | CLIENT SETNAME | - | - | - |
| - | - | - | HELLO | - | - | - |

**Table 6-27** 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 | FLUSHALL<br>(FLUSHALL SYNC not supported.) |
| DUMP | BITCOUNT | HEXISTS | BRPOP | SCARD | ZCARD | FLUSHDB |
| EXISTS | BITOP | HGET | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | HGETALL | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | HINCRBY | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | HINCRBYFLOAT | LLEN | SINTERSTORE | ZRANGEBYSCORE | MONITOR |
| PTTL | GET | HKEYS | LPOP | SISMEMBER | ZRANK | SLOWLOG |
| RANDOMKEY | GETRANGE | HMGET | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | ROLE |
| RENAME | GETSET | HMSET | LRANGE | SMOVE | ZREMRANGEBYCORE | SWAPDB |
| RENAMENX | INCR | HSET | LREM | SPOP | ZREVRANGE | MEMORY |
| RESTORE | INCRBY | HSETNX | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | COMMAND |
| SORT | INCRBYFLOAT | HVALS | LTRIM | SREM | ZREVRANK | COMMAND COUNT |
| TTL | MGET | HSCAN | RPOP | SUNION | ZSCORE | COMMAND GETKEYS |
| TYPE | MSET | HSTRLEN | RPOPLPUSH | SUNIONSTORE | ZUNIONSTORE | COMMAND INFO |

| Generic (Key) | String | Hash | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|---|
| SCAN | MSETNX | HLEN | RPUSH | SSCAN | ZINTERSTORE | CONFIG GET |
| OBJECT | PSETEX | HRANDFIELD | RPUSHX | SMISMEMBER | ZSCAN | CONFIG RESETSTAT |
| PEXPIRE | SET | - | LPUSH | - | ZRANGEBYLEX | CONFIG REWRITE |
| PEXPIREAT | SETBIT | - | BLMOVE | - | ZLEXCOUNT | CONFIG SET |
| EXPIREAT | SETEX | - | LMOVE | - | ZREMRANGEBYSCORE | - |
| KEYS | SETNX | - | LPOS | - | ZREM | - |
| UNLINK | SETRANGE | - | - | - | ZREMRANGEBYLEX | - |
| TOUCH | STRLEN | - | - | - | BZPOPMAX | - |
| COPY | BITFIELD | - | - | - | BZPOPMIN | - |
| - | GETBIT | - | - | - | ZPOPMAX | - |
| - | BITFIELD_RO | - | - | - | ZPOPMIN | - |
| - | GETDEL | - | - | - | ZREVRANGEBYLEX | - |
| - | GETEX | - | - | - | ZDIFF | - |
| - | - | - | - | - | ZDIFFSTORE | - |
| - | - | - | - | - | ZINTER | - |
| - | - | - | - | - | ZMSCORE | - |
| - | - | - | - | - | ZRANDMEMBER | - |
| - | - | - | - | - | ZRANGESTORE | - |

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

**Table 6-28** Commands supported by read/write splitting DCS Redis 6.0 instances (2)

| HyperLoglog | Pub/Sub | Transactions | Connection | Scripting | Geo | Stream |
|---|---|---|---|---|---|---|
| PFADD | PSUBSCRIBE | DISCARD | AUTH | EVAL | GEOADD | XAUTOCLAIM |
| PFCOUNT | PUBLISH | EXEC | ECHO | EVALSHA | GEOHASH | XGROUP CREATECONSUMER |
| PFMERGE | PUBSUB | MULTI | PING | SCRIPT EXISTS | GEOPOS | XACK |
| - | PUNSUBSCRIBE | UNWATCH | QUIT | SCRIPT FLUSH | GEODIST | XADD |
| - | SUBSCRIBE | WATCH | SELECT | SCRIPT KILL | GEORADIUS | XCLAIM |
| - | UNSUBSCRIBE | - | CLIENT KILL | SCRIPT LOAD | GEORADIUSBYMEMBER | XDEL |
| - | - | - | CLIENT LIST | SCRIPT DEBUG YES\|SYNC\|NO | GEOSEARCH | XGROUP |
| - | - | - | CLIENT GETNAME | - | GEOSEARCHSTORE | XINFO |
| - | - | - | CLIENT SETNAME | - | - | XLEN |
| - | - | - | HELLO | - | - | XPENDING |
| - | - | - | - | - | - | XRANGE |
| - | - | - | - | - | - | XREAD |
| - | - | - | - | - | - | XREADGROUP |

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

## Commands Disabled by DCS for Redis 6.0

**Table 6-29** 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 |
| - | - | CLUSTER RESET |

**Table 6-30** 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 |

| Generic (Key) | Server | Connection |
| --- | --- | --- |
| - | DEBUG SEGFAULT | CLIENT TRACKINGINFO |
| - | LASTSAVE | CLIENT UNPAUSE |
| - | PSYNC | RESET |
| - | SAVE | - |
| - | SHUTDOWN | - |
| - | SLAVEOF | - |
| - | LATENCY commands | - |
| - | MODULE commands | - |
| - | LOLWUT | - |
| - | SWAPDB | - |
| - | REPLICAOF | - |
| - | SYNC | - |
| - | ACL | - |
| - | FAILOVER | - |

**Table 6-31** 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 |
| - | DEBUG SEGFAULT | CLIENT TRACKING |
| - | LASTSAVE | CLIENT TRACKINGINFO |
| - | LOLWUT | CLIENT UNPAUSE |
| - | MODULE LIST/LOAD/ UNLOAD | RESET |
| - | PSYNC | - |
| - | REPLICAOF | - |
| - | SAVE | - |
| - | SHUTDOWN [NOSAVE\| SAVE] | - |

| Generic | Server | Connection |
| --- | --- | --- |
| - | SLAVEOF | - |
| - | SWAPDB | - |
| - | SYNC | - |
| - | ACL | - |
| - | FAILOVER | - |

### Commands That Can Be Renamed

**Table 6-32** Commands that can be renamed

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

## 6.4 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.

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 6-33** Commands supported by Web CLI (1)

| Generic (Key) | String | List | Set | Sorted Set | Server |
| --- | --- | --- | --- | --- | --- |
| DEL | APPEND | RPUSH | SADD | ZADD | FLUSHALL |
| OBJECT | BITCOUNT | RPUSHX | SCARD | ZCARD | FLUSHDB |

| Generic (Key) | String | List | Set | Sorted Set | Server |
|---|---|---|---|---|---|
| EXISTS | BITOP | BRPOPLRUSH | SDIFF | ZCOUNT | DBSIZE |
| EXPIRE | BITPOS | LINDEX | SDIFFSTORE | ZINCRBY | TIME |
| MOVE | DECR | LINSERT | SINTER | ZRANGE | INFO |
| PERSIST | DECRBY | LLEN | SINTERSTORE | ZRANGEBYSCORE | CLIENT KILL |
| PTTL | GET | LPOP | SISMEMBER | ZRANK | CLIENT LIST |
| RANDOMKEY | GETRANGE | LPUSHX | SMEMBERS | ZREMRANGEBYRANK | CLIENT GETNAME |
| RENAME | GETSET | LRANGE | SMOVE | ZREMRANGEBYCORE | CLIENT SETNAME |
| RENAMENX | INCR | LREM | SPOP | ZREVRANGE | CONFIG GET |
| SCAN | INCRBY | LSET | SRANDMEMBER | ZREVRANGEBYSCORE | SLOWLOG |
| SORT | INCRBYFLOAT | LTRIM | SREM | ZREVRANK | ROLE |
| TTL | MGET | RPOP | SUNION | ZSCORE | SWAPDB |
| TYPE | MSET | RPOPLPU | SUNIONSTORE | ZUNIONSTORE | MEMORY |
| - | MSETNX | RPOPLPUSH | SSCAN | ZINTERSTORE | - |
| - | PSETEX | - | - | ZSCAN | - |
| - | SET | - | - | ZRANGEBYLEX | - |
| - | SETBIT | - | - | ZLEXCOUNT | - |
| - | SETEX | - | - | - | - |
| - | SETNX | - | - | - | - |
| - | SETRANGE | - | - | - | - |
| - | STRLEN | - | - | - | - |
| - | BITFIELD | - | - | - | - |

**Table 6-34** Commands supported by Web CLI (2)

| Hash | HyperLoglog | Connection | Scripting | Geo | Pub/Sub |
|---|---|---|---|---|---|
| HDEL | PFADD | AUTH | EVAL | GEOADD | UNSUBSCRIBE |
| HEXISTS | PFCOUNT | ECHO | EVALSHA | GEOHASH | PUBLISH |
| HGET | PFMERGE | PING | SCRIPT EXISTS | GEOPOS | PUBSUB |
| HGETALL | - | QUIT | SCRIPT FLUSH | GEODIST | PUNSUBSCRIBE |
| HINCRBY | - | - | SCRIPT KILL | GEORADIUS | - |
| HINCRBYFLOAT | - | - | SCRIPT LOAD | GEORADIUS BYMEMBER | - |
| HKEYS | - | - | - | - | - |
| HMGET | - | - | - | - | - |
| HMSET | - | - | - | - | - |
| HSET | - | - | - | - | - |
| HSETNX | - | - | - | - | - |
| HVALS | - | - | - | - | - |
| HSCAN | - | - | - | - | - |
| HSTRLEN | - | - | - | - | - |

## Commands Disabled in Web CLI

The following lists the commands disabled when the using Web CLI.

**Table 6-35** 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 |
| - | CONFIG REWRITE | MULTI | CLUSTER SETSLOT |

| Generic (Key) | Server | Transactions | Cluster |
|---|---|---|---|
| - | CONFIG RESETSTAT | WATCH | CLUSTER BUMPEPOCH |
| - | SAVE | - | CLUSTER SAVECONFIG |
| - | BGSAVE | - | CLUSTER FORGET |
| - | BGREWRITEAOF | - | CLUSTER REPLICATE |
| - | COMMAND | - | CLUSTER COUNT-FAILURE-REPORTS |
| - | KEYS | - | CLUSTER FAILOVER |
| - | MONITOR | - | CLUSTER SET-CONFIG-EPOCH |
| - | SYNC | - | CLUSTER RESET |
| - | PSYNC | - | - |
| - | ACL | - | - |
| - | MODULE | - | - |

**Table 6-36** Commands disabled in Web CLI (2)

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

# 6.5 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 6-37**.

Some commands support multiple keys but do not support cross-slot access. For details, see **Table 6-39**. Restricted commands are listed in **Table 6-38**.

**Table 6-40** 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 6-37** 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 *destination*. |
| 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 *destination*. |
| 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 *destination*. |
| SMOVE | Moves **member** from the set at **source** to the set at *destination*. |
| **Sorted Set** | |
| ZUNIONSTORE | Computes the union of *numkeys* sorted sets given by the specified keys. |
| ZINTERSTORE | Computes the intersection of *numkeys* sorted sets given by the specified keys. |
| **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. |
| **Key** | |
| RENAME | Renames *key* to *newkey*. |
| RENAMENX | Renames *key* to *newkey* if *newkey* does not yet exist. |

| Category | Description |
| --- | --- |
| 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 *destination*. |
| **String** | |
| MSETNX | Sets the given keys to their respective values. |

## Redis Commands Restricted in Proxy Cluster DCS Instances

**Table 6-38** 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 sets | BZPOPMAX | For a Proxy Cluster instance, all keys transferred must be in the same slot.<br>Not supported for Proxy Cluster DCS Redis 4.0 instances. |
| | BZPOPMIN | |
| Geo | GEORADIUS | • For a Proxy Cluster instance, all keys transferred must be in the same slot.<br>• For a Proxy Cluster instance with multiple databases, the **STORE** option is not supported. |
| | GEORADIUSBYMEMBER | |
| | GEOSEARCHSTORE | |
| Connection | CLIENT KILL | • Only the following two formats are supported:<br>– CLIENT KILL ip:port<br>– CLIENT KILL ADDR ip:port<br>• The **id** field has a random value, and it does not meet the idc1<idc2→Tc1<Tc2 requirement. |

| Category | Command | Restriction |
| --- | --- | --- |
| | CLIENT LIST | • Only the following two formats are supported:<br>– CLIENT LIST<br>– CLIENT LIST [TYPE normal\|master\|replica\|pubsub]<br>• The **id** field has a random value, and it does not meet the idc1<idc2→Tc1<Tc2 requirement. |
| | SELECT index | Multi-DB of Proxy Cluster instances can be implemented by changing the keys. This solution is not recommended.<br>For details about multi-DB restrictions on Proxy Cluster instances, see **What Are the Constraints on Implementing Multiple Databases on a Proxy Cluster Instance?** |
| HyperLogLog | PFCOUNT | For a Proxy Cluster instance, all keys transferred must be in the same slot. |
| | PFMERGE | |
| Keys | RENAME | For a Proxy Cluster instance, all keys transferred must be in the same slot. |
| | RENAMENX | |
| | SCAN | • Proxy Cluster instances do not support the **SCAN** command in pipelines. |
| Lists | BLPOP | For a Proxy Cluster instance, all keys transferred must be in the same slot. |
| | BRPOP | |
| | 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 keys transferred must be in the same slot.<br>• 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. |
| | EVALSHA | |

| Category | Command | Restriction |
| --- | --- | --- |
| Server | MEMORY DOCTOR | For a Proxy Cluster instance, add the *ip:port* of the node at the end of the command.<br>Do as follows to obtain the IP address and port number of a node (**MEMORY USAGE** is used as an example):<br>1. Run the **cluster keyslot** *key* command to query the slot number of a key.<br>2. Run the **icluster nodes** command to query the IP address and port number corresponding to the slot where the key is.<br>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.<br>3. Run the **MEMORY USAGE** *key ip:port* command.<br>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.<br>The following is an example for a single-DB Proxy Cluster instance:<br>set key1 value1<br>OK<br>get key1<br>value1<br>cluster keyslot key1<br>9189<br>icluster nodes<br>xxx 192.168.00.00:1111@xxx xxx connected 10923-16383<br>xxx 192.168.00.01:2222@xxx xxx connected 0-5460<br>xxx 192.168.00.02:3333@xxx xxx connected 5461-10922<br>MEMORY USAGE key1 192.168.00.02:3333<br>54 |
| | MEMORY HELP | |
| | MEMORY MALLOC-STATS | |
| | MEMORY PURGE | |
| | MEMORY STATS | |
| | MEMORY USAGE | |
| | MONITOR | |

| Category | Command | Restriction |
| --- | --- | --- |
| Strings | BITOP | For a Proxy Cluster instance, all keys transferred must be in the same slot. |
| | MSETNX | |
| Transactions | WATCH | For a Proxy Cluster instance, all keys transferred must be in the same slot. |
| | MULTI | The order of cross-slot commands in a transaction is not guaranteed.<br>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** |
| | EXEC | |
| Streams | XACK | Currently, Proxy Cluster instances do not support Streams. |
| | XADD | |
| | XCLAIM | |
| | XDEL | |
| | XGROUP | |
| | XINFO | |
| | XLEN | |
| | XPENDING | |
| | XRANGE | |
| | XTRIM | |
| | XREVRANGE | |
| | XREAD | |
| | XREADGROUP GROUP | |
| | XAUTOCLAIM | |

### Multi-Key Commands of Proxy Cluster Instances

**Table 6-39** 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 6-40** Redis commands restricted for read/write splitting instances

| Category | Command | Restriction |
| --- | --- | --- |
| Connection | CLIENT KILL | • Only the following two formats are supported:<br>– CLIENT KILL ip:port<br>– CLIENT KILL ADDR ip:port<br>• The **id** field has a random value, and it does not meet the idc1<idc2→Tc1<Tc2 requirement. |
| | CLIENT LIST | • Only the following two formats are supported:<br>– CLIENT LIST<br>– CLIENT LIST [TYPE normal\|master\|replica\|pubsub]<br>• The **id** field has a random value, and it does not meet the idc1<idc2→Tc1<Tc2 requirement. |

## 6.6 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,

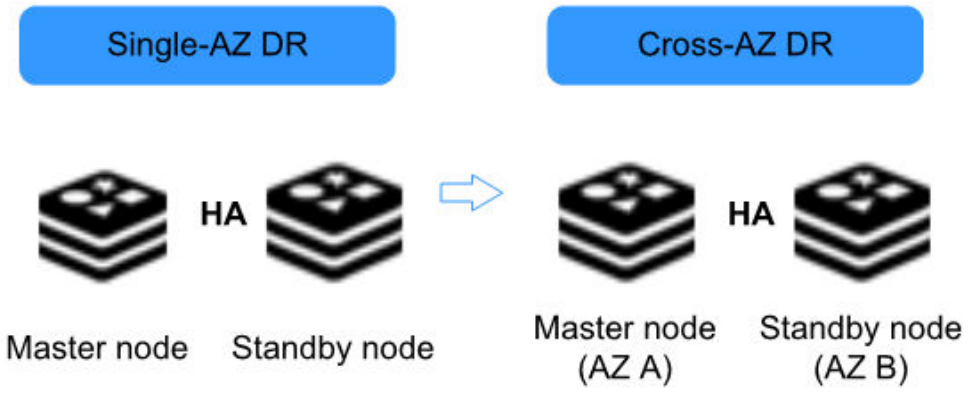
RENAME, RENAMENX, RPOPLPUSH, MSETNX, SCRIPT LOAD, SCRIPT KILL, SCRIPT EXISTS, and SCRIPT FLUSH

# 7 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 7-1** 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.

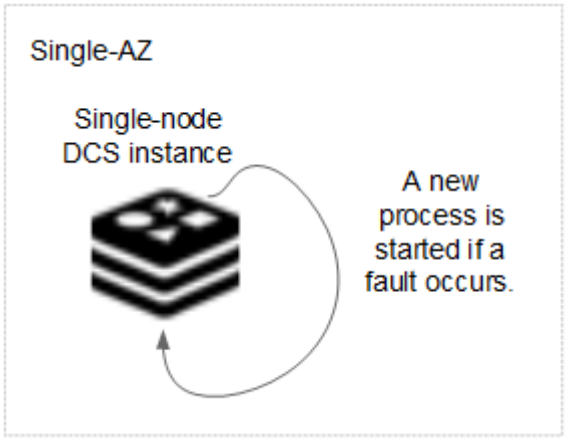
## Single-AZ HA Within a Region

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.

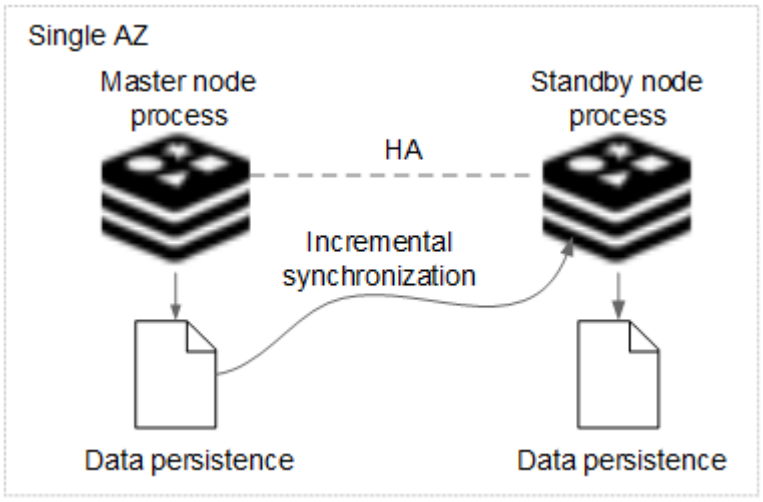
**Figure 7-2** HA for a single-node DCS instance deployed within an AZ




**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 7-3** 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 7-4** Cross-AZ deployment of a master/standby DCS instance

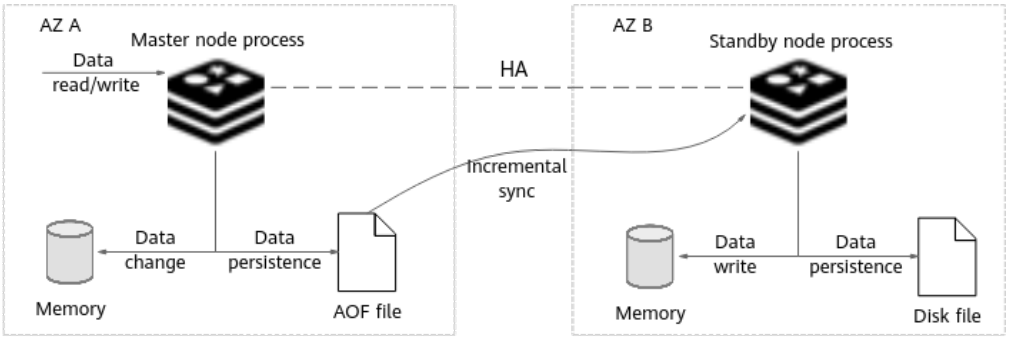



**Figure 7-5** Cross-AZ deployment of a read/write splitting DCS instance

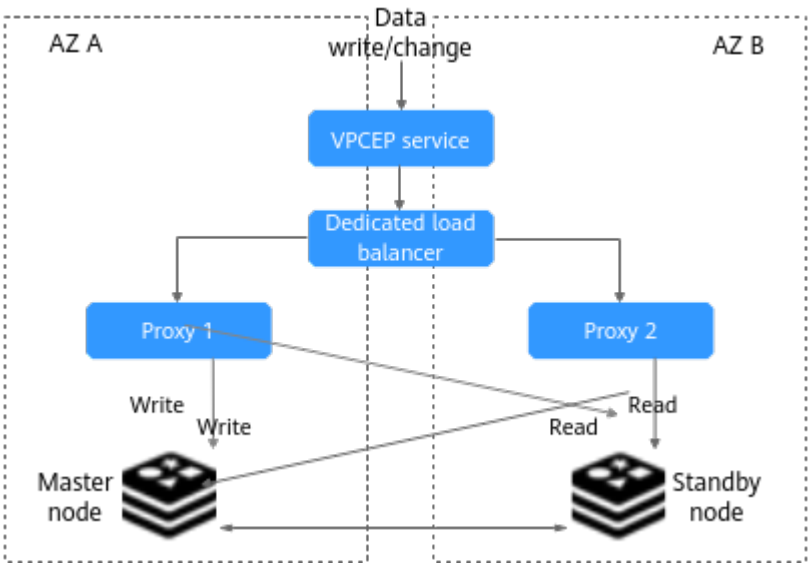


**Figure 7-6** Cross-AZ deployment of a Proxy Cluster DCS instance

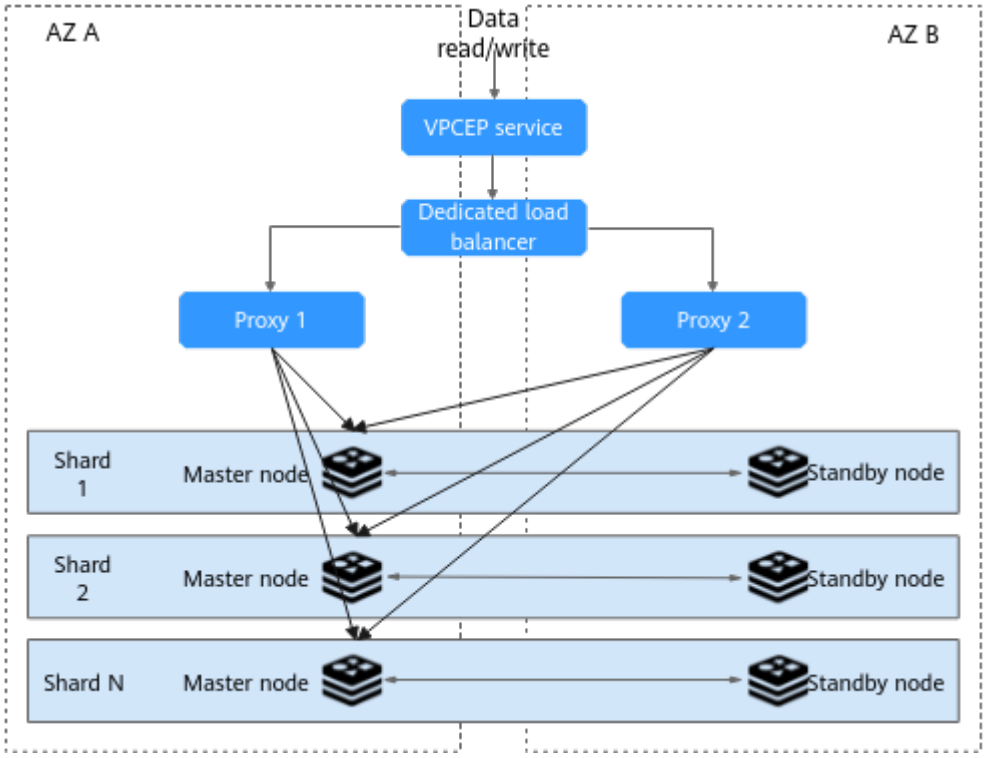
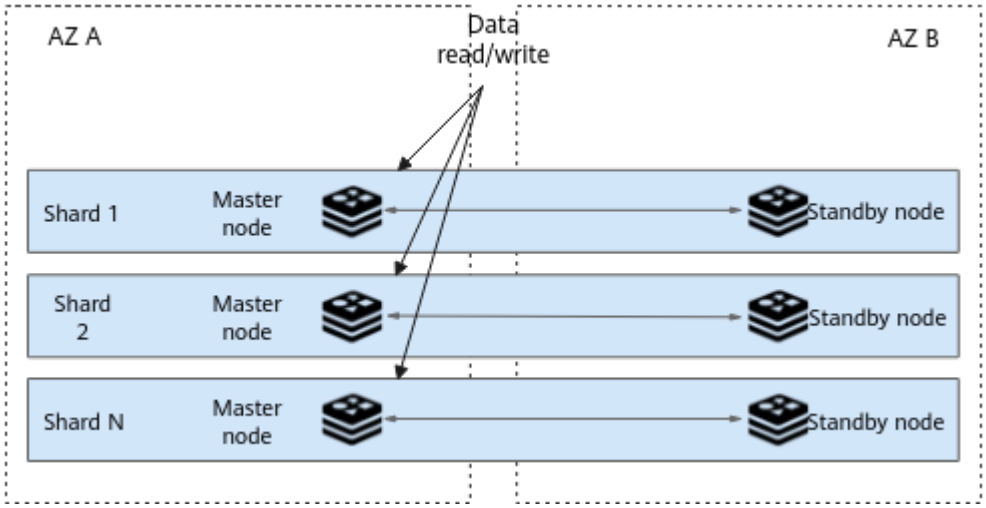


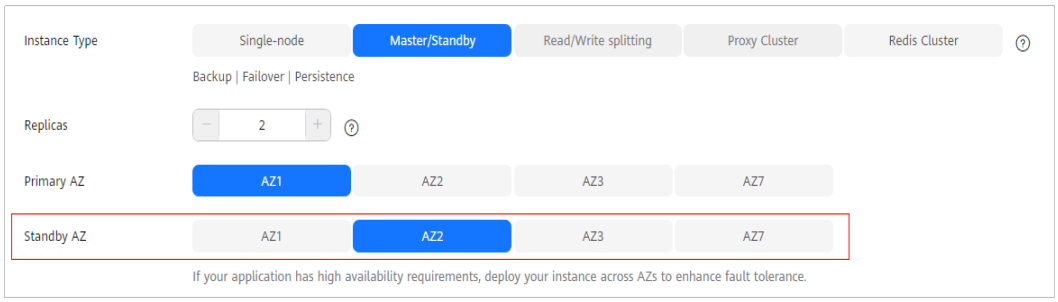

**Figure 7-7** Cross-AZ deployment of a Redis Cluster DCS instance




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.

**Figure 7-8** Selecting different AZs




**NOTE**

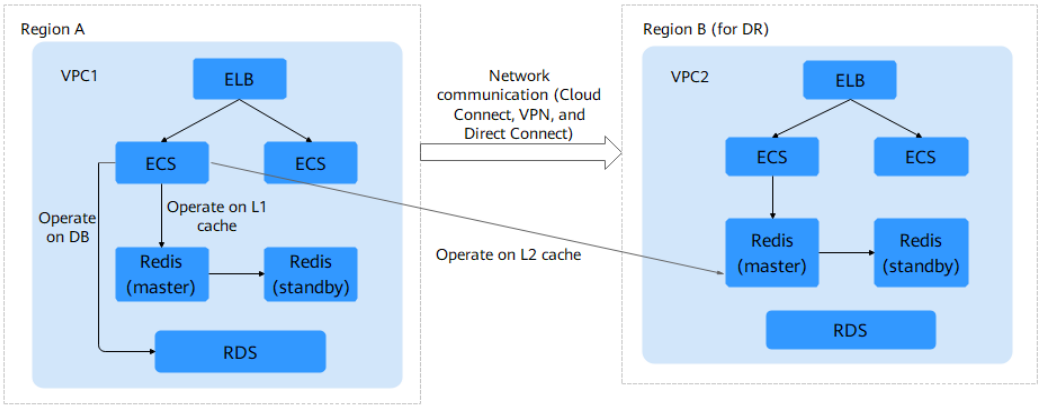
You can also deploy your application across AZs to ensure both data reliability and service availability in the event of power supply or network disruptions.

## 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 7-9** Dual-write on the application side to achieve multi-active




Note:

1. 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.

# 8 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 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 8-1** Differences between DCS for Redis and open-source Redis

| Feature | Open-Source Redis | DCS for Redis |
| --- | --- | --- |
| Service deployment | Requires 0.5 to 2 days to prepare servers. | Container-based deployment in 8s. |
| Version | - | Deeply engaged in the open-source community and supports the latest Redis version. Redis 4.0, 5.0, and 6.0 are supported. |
| Security | Network and server safety is the user's responsibility. | • Network security is ensured using HUAWEI CLOUD VPCs and security groups.<br>• Data reliability is ensured by data replication and scheduled backup. |
| Performance | - | 100,000 QPS per node |

| Feature | Open-Source Redis | DCS for Redis |
| --- | --- | --- |
| Monitoring | Provides only basic statistics. | Provides more than 30 monitoring metrics and customizable alarm threshold and policies.<br>● Various metrics<br>– External metrics include the number of commands, concurrent operations, connections, clients, and denied connections.<br>– Resource usage metrics include CPU usage, physical memory usage, network input throughput, and network output throughput.<br>– Internal metrics include instance capacity usage, as well as the number of keys, expired keys, PubSub channels, PubSub patterns, keyspace hits, and keyspace misses.<br>● Custom alarm thresholds and policies for different metrics to help identify service faults. |
| Backup and restoration | Supported | ● Supports scheduled and manual backup. Backup files can be downloaded.<br>● Backup data can be restored on the console. |
| Parameter management | No visualized parameter management | ● Visualized parameter management is supported on the console.<br>● Configuration parameters can be modified online.<br>● Data can be accessed and modified on the console. |
| 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.<br>● 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 |

# 9 Notes and Constraints

## Redis Instance

**Table 9-1** Notes and constraints

| Item | Notes and Constraints |
| --- | --- |
| Instance version | Currently, DCS supports Redis 4.0, 5.0, and 6.0.<br>Redis instances cannot be upgraded, but data of an earlier instance can be migrated to a later one. |
| Data persistence | • Not available for single-node instances.<br>• 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.<br>• Redis Cluster and master/standby instances: Read/Write splitting is implemented on the client, which requires manual configurations.<br>• 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 9-2** Notes and constraints

| Item | Notes and Constraints |
| --- | --- |
| Changing Redis instance specifications or types | • You are advised to change instances during off-peak hours. Otherwise, the change may fail.<br>• Change the replica quantity and capacity separately.<br>• Only one replica can be deleted per operation.<br>• For more information, see **Modifying Specifications**. |

## Data Migration

**Table 9-3** 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.<br>• You cannot use public networks for online migration.<br>• The source must be Redis 3.0 or later.<br>• 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.<br>• Unavailable for Redis Cluster instances. |

# 10 Billing

DCS supports the pay-per-use mode. For details, see **Product Pricing Details**.

## 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

- 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.

## Configuration Changes

You can change the specifications of a DCS Redis 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**.

## FAQ

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

# 11 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 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 11-1** System-defined roles and policies supported by DCS

| Role/Policy Name | Description | Type | Dependency |
| --- | --- | --- | --- |
| DCS FullAccess | All permissions for DCS. Users granted these permissions can operate and use all DCS instances. | System-defined policy | None |
| DCS UserAccess | Common user permissions for DCS, excluding permissions for creating, modifying, deleting DCS instances and modifying instance specifications. | System-defined policy | None |
| DCS ReadOnlyAccess | 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 | Type | Dependency |
| --- | --- | --- | --- |
| DCS AgencyAccess | Permissions to assign to DCS agencies.<br>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.<br>• Querying a subnet<br>• Querying the subnet list<br>• Querying a port<br>• Querying the port list<br>• Updating a port<br>• Creating a port | System-defined policy | None |

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 11-2** Common operations supported by each system policy

| Operation | DCS FullAccess | DCS UserAccess | DCS ReadOnlyAccess | DCS Administrator |
| --- | --- | --- | --- | --- |
| Modifying instance configuration parameters | √ | √ | × | √ |
| Deleting background tasks | √ | √ | × | √ |
| Accessing instances using Web CLI | √ | √ | × | √ |

| Operation | DCS FullAccess | DCS UserAccess | DCS ReadOnlyAccess | DCS Administrator |
|---|---|---|---|---|
| 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 ReadOnlyAccess | DCS Administrator |
|---|---|---|---|---|
| Downloading instance backup data | √ | √ | × | √ |
| Deleting DCS instances | √ | × | × | √ |
| Querying instance configuration parameters | √ | √ | √ | √ |
| Querying instance restoration logs | √ | √ | √ | √ |
| Querying instance backup logs | √ | √ | √ | √ |
| Querying DCS instances | √ | √ | √ | √ |
| Querying instance background tasks | √ | √ | √ | √ |
| Querying all instances | √ | √ | √ | √ |
| Operating slow queries | √ | √ | √ | √ |

## Helpful Links

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

# 12 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 4.0/5.0/6.0 instance can be accessed in public using Elastic Load Balance (ELB).

## 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

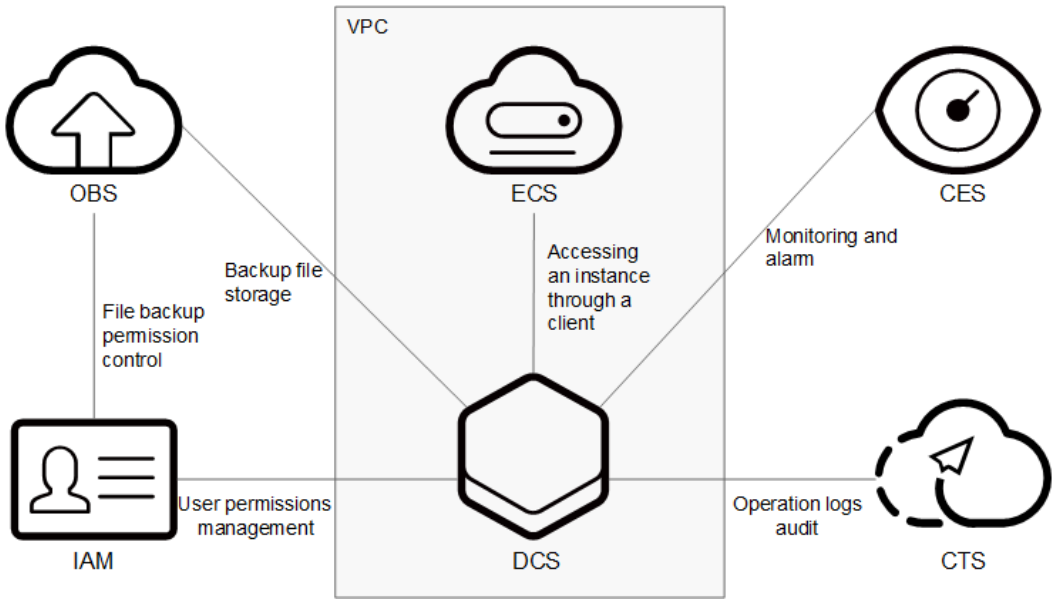
A replica is a **node** of a DCS instance. A single-replica instance has no standby node. A two-replica instance has one master node and one standby node. By default, each master/standby instance has two replicas. If the number of replicas is set to three for a master/standby instance, the instance has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

# 13 Related Services

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

**Figure 13-1** Relationships between DCS and other services




## VPC

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 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.

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.