

Distributed Cache Service

User Guide

Issue 01
Date 2024-06-20



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

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

Trademarks and Permissions



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

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

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

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

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

Contents

1 Before You Start.....	1
1.1 Accessing and Using DCS.....	1
1.2 Using the DCS Console.....	2
2 Permissions Management.....	4
2.1 Creating a User and Granting DCS Permissions.....	4
2.2 DCS Custom Policies.....	5
2.3 DCS Resources.....	6
3 Buying a DCS Instance.....	8
3.1 Identifying Requirements.....	8
3.2 Preparing Required Resources.....	9
3.3 Buying a DCS Redis Instance.....	9
4 Accessing a DCS Redis Instance.....	13
4.1 Restrictions.....	13
4.2 redis-cli.....	13
4.3 Access in Different Languages.....	18
4.3.1 Java.....	18
4.3.1.1 Jedis.....	18
4.3.1.2 Lettuce.....	25
4.3.1.3 Redisson.....	37
4.3.2 Clients in Python.....	47
4.3.3 go-redis.....	50
4.3.4 hiredis in C++.....	51
4.3.5 C#.....	54
4.3.6 PHP.....	56
4.3.6.1 phpredis.....	56
4.3.6.2 Predis.....	58
4.3.7 Node.js.....	59
4.4 Connecting to Redis on the Console.....	62
5 Operating DCS Instances.....	64
5.1 Viewing Instance Details.....	64
5.2 Modifying Specifications.....	66
5.3 Restarting an Instance.....	72

5.4 Deleting an Instance.....	73
5.5 Performing a Master/Standby Switchover.....	74
5.6 Clearing DCS Instance Data.....	75
5.7 Exporting Instance List.....	76
5.8 Renaming Commands.....	76
6 Managing DCS Instances.....	78
6.1 Configuration Notice.....	78
6.2 Modifying Configuration Parameters.....	79
6.2.1 Modifying Configuration Parameters of an Instance.....	79
6.3 Modifying Maintenance Window.....	88
6.4 Viewing Background Tasks.....	88
6.5 Managing IP Address Whitelist.....	89
6.6 Managing Tags.....	90
6.7 Managing Nodes.....	92
6.8 Cache Analysis.....	94
6.8.1 Analyzing Big Keys and Hot Keys.....	94
6.8.2 Scanning Expired Keys.....	97
6.9 Viewing Redis Slow Queries.....	102
6.10 Viewing Redis Run Logs.....	103
6.11 Managing Users.....	104
6.12 Diagnosing an Instance.....	105
7 Backing Up and Restoring Instances.....	107
7.1 Overview.....	107
7.2 Configuring an Automatic Backup Policy.....	109
7.3 Manually Backing Up a DCS Instance.....	111
7.4 Restoring a DCS Instance.....	112
7.5 Downloading an RDB or AOF Backup File.....	113
8 Migrating Instance Data.....	115
8.1 Data Migration Overview.....	115
8.2 Importing Backup Files from an OBS Bucket.....	116
8.3 Importing Backup Files from Redis.....	118
8.4 Online Migration.....	119
8.5 IP Switching.....	123
9 Parameter Templates.....	127
9.1 Viewing Parameter Templates.....	127
9.2 Creating a Custom Parameter Template.....	135
9.3 Modifying a Custom Parameter Template.....	145
9.4 Deleting a Custom Parameter Template.....	154
10 Managing Passwords.....	155
10.1 DCS Instance Passwords.....	155

10.2 Changing Instance Passwords.....	156
10.3 Resetting Instance Passwords.....	157
10.4 Changing Password Settings for DCS Redis Instances.....	158
11 Quotas.....	159
12 Monitoring.....	161
12.1 DCS Metrics.....	161
12.2 Common Metrics.....	183
12.3 Viewing Metrics.....	185
12.4 Configuring Alarm Rules for Critical Metrics.....	185
13 Auditing.....	194
13.1 Operations Logged by CTS.....	194
13.2 Querying Real-Time Traces.....	198

1 Before You Start

1.1 Accessing and Using DCS

Accessing DCS

You can access Distributed Cache Service (DCS) from the web-based management console or by using RESTful application programming interfaces (APIs) through HTTPS requests.

- Using the management console

Log in to the management console and choose **Distributed Cache Service** from the service list.

For details on how to use the DCS console, see chapters from [Buying a DCS Instance](#) to [Managing Passwords](#).

DCS monitoring data is recorded by Cloud Eye. To view the monitoring metrics or configure alarm rules, go to the Cloud Eye console. For details, see [Viewing Metrics](#).

If you have enabled Cloud Trace Service (CTS), DCS instance operations are recorded by CTS. You can view the operations history on the CTS console. For details, see [Querying Real-Time Traces](#).

- Using APIs

DCS provides RESTful APIs for you to integrate DCS into your own application system. For details about DCS APIs and API calling, see the [Distributed Cache Service API Reference](#).

NOTICE

1. All available functions can be used on the console. Some functions can also be used through APIs. For more information on how to use functions through APIs, see the [Distributed Cache Service API Reference](#).
 2. For details about APIs for monitoring and auditing, see the [Cloud Eye](#) and [Cloud Trace Service \(CTS\)](#) documentation.
-

Using DCS

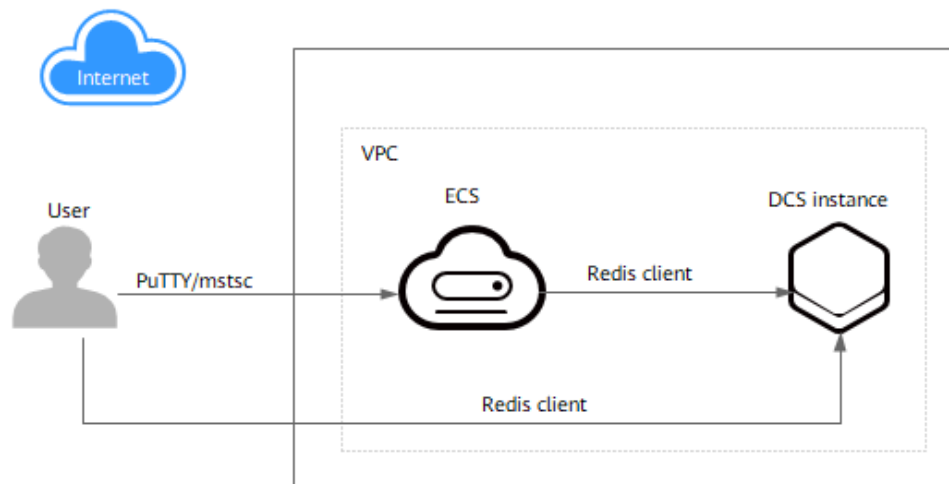
After creating a DCS instance, access it by referring to [Accessing a DCS Redis Instance](#). Any client that is compatible with the open-source Redis protocol can respectively access a DCS Redis instance. After accessing a DCS instance, you can enjoy the fast read/write operations enabled by DCS.

NOTICE

DCS does not involve sensitive user information. Which, why, when, and how data is processed with DCS must comply with local laws and regulations. If sensitive data needs to be transmitted or stored, encrypt data before transmission or storage.

For details on how to access a DCS instance, see the following figure.

Figure 1-1 Accessing a DCS instance



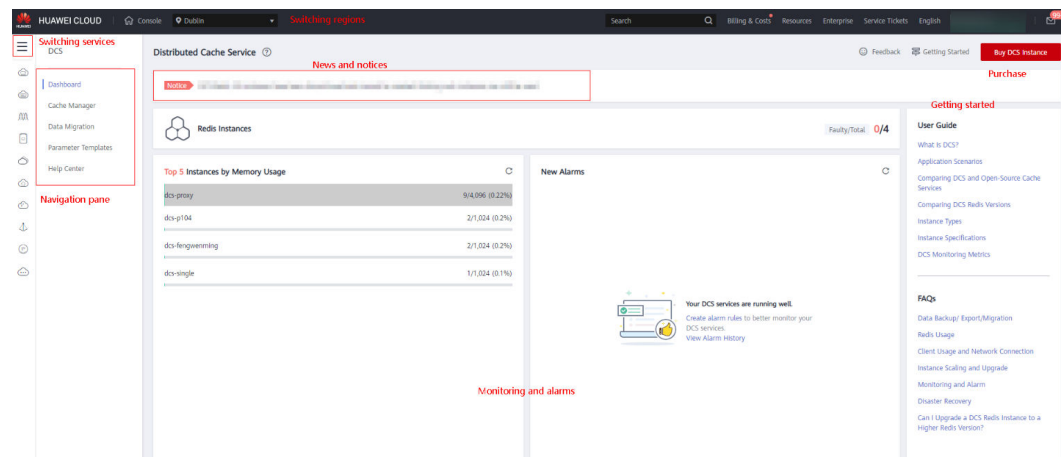
NOTE

- Currently, a DCS instance can be accessed over an internal network through an Elastic Cloud Server (ECS) that is in the same Virtual Private Cloud (VPC) as the DCS instance.

1.2 Using the DCS Console

On the DCS console, you can buy, use, and maintain DCS instances, view instance status and memory usage, and seek online support.

Figure 1-2 DCS console



- **Switching regions**
You can switch to a region closer to your application.
- **Switching services**
You can switch to consoles of other services, such as the VPC and Cloud Eye consoles.
- **Creating an instance**
Click to buy DCS Redis instances.
- **Navigation pane**
This area provides access to operating DCS instances and migrating data.
- **News and notices**
This area informs you of the latest available features and special offers.
- **Instances**
This area displays the total number of instances and the number of faulty instances of the current user.
- **Monitoring and alarms**
This area displays instances with the highest memory usage. For details on how to view information about a specific instance, see [Viewing Instance Details](#).
You can create alarm rules for your instance. When an alarm is generated, you can handle it immediately. For details, see [Configuring Alarm Rules for Critical Metrics](#).
- **Getting started**
By clicking these links, you will be directed to the documentation to learn more about how to use DCS.
- **Online support**
If you have any questions while using DCS, contact online support.

2 Permissions Management

2.1 Creating a User and Granting DCS Permissions

This section describes how to use [Identity and Access Management \(IAM\)](#) to implement fine-grained permissions control for your DCS resources. With IAM, you can:

- Create IAM users for employees based on your enterprise's organizational structure. Each IAM user will have their own security credentials for accessing DCS resources.
- Manage permissions on a principle of least permissions (PoLP) basis.
- Entrust a Huawei Cloud account or cloud service to perform efficient O&M on your DCS resources.

If your Huawei Cloud account does not require individual IAM users, skip this chapter.

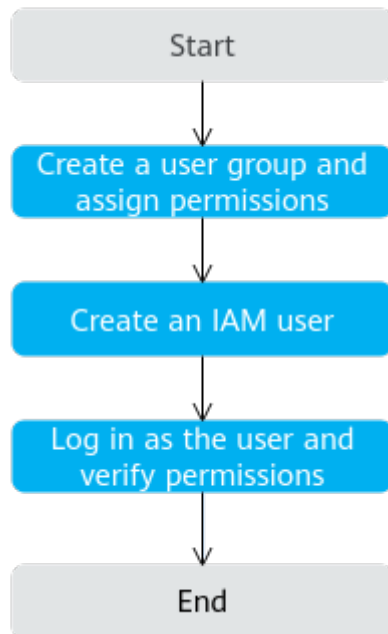
This section describes the procedure for granting the **DCS ReadOnlyAccess** permission (see [Figure 2-1](#)) as an example.

Prerequisites

Learn about the permissions (see [System-defined roles and policies supported by DCS](#)) supported by DCS and choose policies or roles according to your requirements. For the permissions of other services, see [Permissions Policies](#).

Process Flow

Figure 2-1 Process of granting DCS permissions



1. **Create a user group and assign permissions.**
Create a user group on the IAM console, and assign the **DCS ReadOnlyAccess** policy to the group.
2. **Create a user and add it to a user group.**
Create a user on the IAM console and add the user to the group created in 1.
3. **Log in** and verify permissions.
Log in to the DCS console using the newly created user, and verify that the user only has read permissions for DCS.
 - Choose **Distributed Cache Service** in **Service List**. Then click **Buy DCS Instance** in the upper right corner of the DCS console. If a DCS instance cannot be purchased, the **DCS ReadOnlyAccess** policy has already taken effect.
 - Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to access the service, the **DCS ReadOnlyAccess** policy has already taken effect.

2.2 DCS Custom Policies

Custom policies can be created to supplement the system-defined policies of DCS. For the actions that can be added to custom policies, see [Permissions Policies and Supported Actions](#).

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.

- JSON: Edit JSON policies from scratch or based on an existing policy.

For details, see [Creating a Custom Policy](#). The following section contains examples of common DCS custom policies.

NOTE

Due to data caching, a policy involving OBS actions will take effect five minutes after it is attached to a user, user group, or project.

Example Custom Policies

- Example 1: Allowing users to delete and restart DCS instances and clear data of an instance

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dcs:instance:delete",
        "dcs:instance:modifyStatus"
      ]
    }
  ]
}
```

- Example 2: Denying DCS instance deletion

A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

For example, if you want to assign all of the permissions of the **DCS FullAccess** policy to a user, except for deleting DCS instances, you can create a custom policy to deny only DCS instance deletion. When you apply both the **DCS FullAccess** policy and the custom policy denying DCS instance deletion, since "Deny" always takes precedence over "Allow", the "Deny" will be applied for that one conflicting permission. The user will then be able to perform all operations on DCS instances except deleting DCS instances. The following is an example of a deny policy:

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "dcs:instance:delete"
      ]
    }
  ]
}
```

2.3 DCS Resources

A resource is an object that exists within a service. DCS resources are instances. You can select them by specifying their paths.

Table 2-1 DCS resources and their paths

Resource	Resource Name	Path
instance	Instance	[Format] DCS:*:*:instance: <i>instance ID</i> [Note] For instance resources, DCS automatically generates the prefix (DCS:*:*:instance:) of the resource path. For the path of a specific instance, add the <i>instance ID</i> to the end. You can also use an asterisk * to specify any instance. For example: DCS:*:*:instance:* indicates any DCS instance.

3 Buying a DCS Instance

3.1 Identifying Requirements

Before purchasing a DCS instance, identify your requirements:

1. Decide on the instance type.

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

2. Decide on the required instance specification.

Each specification specifies the maximum available memory, number of connections, and bandwidth. For details, see [DCS Instance Specifications](#).

3. Decide on the region and whether cross-AZ deployment is required.

Choose a region closest to your application to reduce latency.

A region consists of multiple availability zones (AZs) with physically isolated power supplies and networks. Master/standby and cluster DCS instances can be deployed across AZs. Applications can also be deployed across AZs to achieve high availability (HA) for both data and applications.

NOTE

- If a master/standby or cluster DCS instance is deployed across AZs, faults in an AZ do not affect cache nodes in other AZs. This is because when the master node is faulty, the standby cache node will automatically become the master node to provide services. Such deployment achieves better disaster recovery.
 - Deploying a DCS instance across AZs slightly reduces network efficiency compared with deploying an instance within an AZ. Therefore, if a DCS instance is deployed across AZs, synchronization between master and standby cache nodes is slightly less efficient.
4. Decide whether backup policies are required.
Currently, backup policies can be configured only for master/standby, Proxy Cluster, and Redis Cluster DCS instances. For details about backup and restoration, see [Overview](#).

3.2 Preparing Required Resources

Overview

Before creating a DCS instance, prepare the required resources, including a VPC and a subnet. Each DCS instance is deployed in a VPC and bound to a specific subnet and security group, which provide an isolated virtual network environment and security protection policies which you can easily configure and manage.

If you already have a VPC, subnet, and security group, you can use them for all DCS instances you subsequently create.

Required Resources

The following table lists the resources required by a DCS instance.

Table 3-1 Dependency resources of a DCS instance

Resource	Requirement	Operations
VPC and subnet	Different DCS instances can use the same or different VPCs and subnets based on site requirements. Note the following when creating a VPC and subnet: <ul style="list-style-type: none">The VPC and the DCS instance must be in the same region.Retain the default settings unless otherwise specified.	For details on how to create a VPC and subnet, see Creating a VPC . If you need to create and use a new subnet in an existing VPC, see Creating a Subnet for the VPC .

3.3 Buying a DCS Redis Instance

You can buy one or more DCS Redis instances with the required computing capabilities and storage space based on service requirements.

Prerequisites

- To achieve fine-grained management of your HUAWEI CLOUD resources, create IAM user groups and users and grant specified permissions to the users. For details, see [Permission Management](#).
- You have prepared necessary resources.

Procedure

Step 1 Go to the [Buy DCS Instance](#) page.

Step 2 Select a region closest to your application to reduce latency and accelerate access.

Step 3 Set the following information:

1. **Cache Engine:** The default engine is **Redis**.
2. **Version:**

Currently supported Redis versions: 4.0/5.0/6.0

NOTE

- The Redis version cannot be changed once the instance is created. To use a later Redis version, create another DCS Redis instance 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 [Accessing a DCS Redis Instance](#).

3. **Instance Type:** Options include **Single-node**, **Master/Standby**, **Read/Write splitting**, **Proxy Cluster**, and **Redis Cluster**.

4. **Replicas:** The default value is 2.

This parameter is displayed only when the instance type is master/standby, read/write splitting, or Redis Cluster.

5. Select an AZ.

NOTE

To accelerate access, deploy your instance and your application in the same AZ.

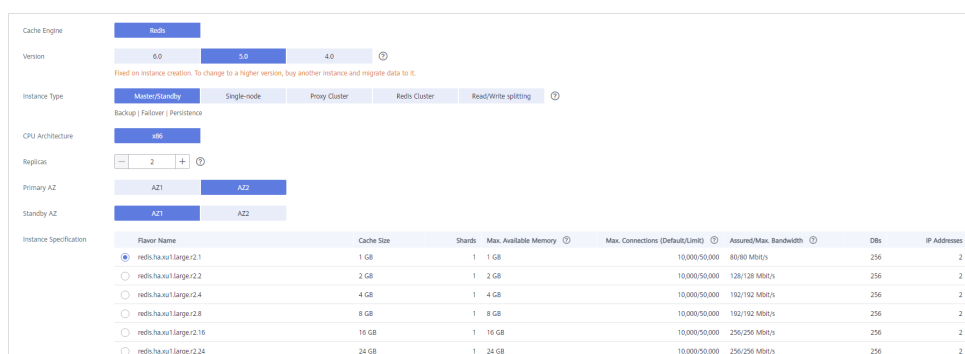
If the instance type is master/standby, read/write splitting, Proxy Cluster, or Redis Cluster, **AZ** becomes **Primary AZ**, and **Standby AZ** is displayed. Select an AZ for the master and standby nodes of the instance.

6. **Instance Specification:**

The default quota is displayed on the console.

The instance parameter settings are shown in the following figure.

Figure 3-1 Buying a DCS Redis instance



Step 4 Configure the instance network parameters.

1. Select a VPC and a subnet.
2. Configure the IP address.

Redis Cluster instances only support automatically-assigned IP addresses. The other instance types support both automatically-assigned IP addresses and manually-specified IP addresses. You can manually specify a private IP address for your instance as required.

For Redis instances, you can specify a port numbering in the range from 1 to 65535. If no port is specified, the default port 6379 will be used.

Step 5 Set the instance password.

- Select **Yes** or **No** for **Password Protected**.

 **NOTE**

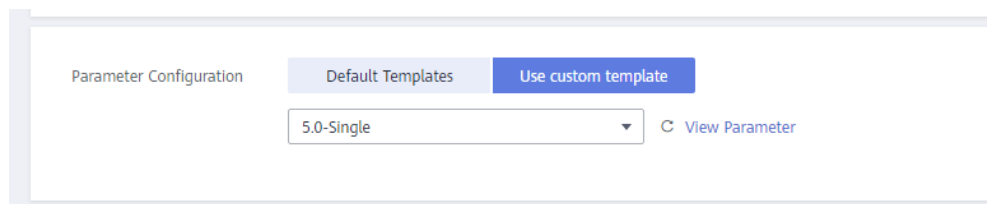
- Password-free access carries security risks. Exercise caution when selecting this mode.
- Even if you created a password-free DCS Redis instance, you can still set a password after the instance is created. For details, see [Changing Password Settings for DCS Redis Instances](#).
- **Password** and **Confirm Password**: These parameters indicate the password of accessing the DCS Redis instance, and are displayed only when **Password Protected** is set to **Yes**.

 **NOTE**

For security purposes, if password-free access is disabled, the system prompts you to enter an instance-specific password when you are accessing the DCS Redis instance. Keep your instance password secure and change it periodically.

Step 6 Configure **Parameter Configuration**.

You can select **Default Templates** or **Use custom template**.



 **NOTE**

On the instance creation page, the default parameter templates are used by default.

Step 7 Specify the backup policy.

This parameter is displayed only when the instance type is master/standby, read/write splitting, or cluster. For details about instance backup and backup policies, see [Backing Up and Restoring Instances](#).

Step 8 Specify the quantity.

Step 9 Enter an instance name and select an enterprise project.

When you create only one instance at a time, the value of **Name** can contain 4 to 64 characters. When you create more than one instance at a time, the value of **Name** can contain 4 to 56 characters. These instances are named in the format of "*name-n*", in which *n* starts from 000 and is incremented by 1. For example, if you create two instances and set **Name** to **dc_demo**, the two instances are respectively named as **dc_demo-000** and **dc_demo-001**.

 **NOTE**

If you cannot select an enterprise project, check your permissions. For details, see [Why Can't I Select the Required Enterprise Project When Creating a DCS Instance?](#)

Step 10 Click **More Settings** to display more configurations, including backup policy and critical command renaming.

1. Enter a description of the instance.
2. Rename critical commands.

Currently, you can only rename the **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, **HGETALL**, **SCAN**, **HSCAN**, **SSCAN**, and **ZSCAN** commands. For Proxy Cluster instances, you can also rename the **DBSIZE** and **DBSTATS** commands.

3. Specify the maintenance window.

Choose a window for DCS O&M personnel to perform maintenance on your instance. You will be contacted before any maintenance activities are performed.

4. Add a tag.

Tags are used to identify cloud resources. When you have many cloud resources of the same type, you can use tags to classify cloud resources by dimension (for example, by usage, owner, or environment).

- If you have created predefined tags, select a predefined pair of tag key and value. Click **View predefined tags**. On the Tag Management Service (TMS) console, view predefined tags or create new tags.
- You can also add a tag by entering the tag key and value. For details about how to name tags, see [Managing Tags](#).

Step 11 Click **Next**.

The displayed page shows the instance information you have specified.

Step 12 Confirm the instance information and submit the request.

Step 13 Return to the **Cache Manager** page to view and manage your DCS instances.

----End

4 Accessing a DCS Redis Instance

4.1 Restrictions

You can access a DCS instance through any Redis client. For details about Redis clients, see the [Redis official website](#).

- To access a DCS Redis instance through a client on an ECS in the same VPC as the instance, note that:
 - The ECS where the Redis client is installed must be in the same VPC as the DCS Redis instance. An ECS and a DCS instance can communicate with each other only when they belong to the same VPC. The IP address of the ECS must be on the whitelist of the DCS instance.
For details about how to configure a whitelist, see [Managing IP Address Whitelist](#).
 - If the ECS and DCS Redis instance are not in the same VPC, connect them by establishing a VPC peering connection. For details, see [Does DCS Support Cross-VPC Access?](#)

4.2 redis-cli

This section describes how to use redis-cli on an ECS in the same VPC as a DCS Redis instance to connect to the instance. For details about more clients, see the [Redis official website](#).

 NOTE

- For Redis 4.0/5.0/6.0, you can specify a port or use the default port 6379. The following uses the default port 6379. If you have specified a port, replace 6379 with the actual port.
- **When connecting to a Redis Cluster instance, ensure that `-c` is added to the command.** Otherwise, the connection will fail.
 - Run the following command to connect to a Redis Cluster instance:
`./redis-cli -h {dcs_instance_address} -p 6379 -a {password} -c`
 - Run the following command to connect to a single-node, master/standby, or Proxy Cluster instance:
`./redis-cli -h {dcs_instance_address} -p 6379 -a {password}`

For details, see the procedure in this section.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

Procedure (Linux)

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Install redis-cli.

The following steps assume that your client is installed on the Linux OS.

1. Log in to the ECS.
2. Run the following command to download the source code package of your Redis client from <https://download.redis.io/releases/redis-6.2.13.tar.gz>:

```
wget http://download.redis.io/releases/redis-6.2.13.tar.gz
```

 NOTE

The following uses redis-6.2.13 as an example. For details, see the [Redis official website](#).

3. Run the following command to decompress the source code package of your Redis client:

```
tar -xzf redis-6.2.13.tar.gz
```

4. Run the following commands to go to the Redis directory and compile the source code of your Redis client:

```
cd redis-6.2.13
```

```
make
```

```
cd src
```

NOTE

If the source code of your Redis client is v6.0 and later, and redis-cli that supports TLS/SSL is required, replace the **make** command with **make BUILD_TLS=yes** to enable TLS.

Step 3 Access the DCS Redis instance.

- Access a DCS instance of a type other than Redis Cluster.

Perform the following procedure to access a single-node, master/standby, read/write splitting, or Proxy Cluster instance.

- a. Run the following command to access the chosen DCS Redis instance:

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

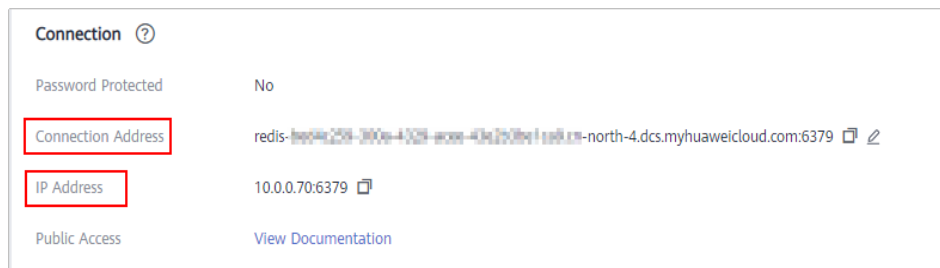
{dcs_instance_address} indicates the IP address/domain name of the DCS instance and **6379** is the port used for accessing the instance. The IP address/domain name and port number are obtained in [Step 1](#).

NOTE

For a Proxy Cluster DCS Redis instance, you can use the **Connection Address** or **IP Address** for *{dcs_instance_address}*. The addresses can be obtained on the instance basic information page on the console, as shown in [Figure 4-1](#).

- **Connection Address** and **IP Address** are the LB addresses. Requests are distributed across proxy nodes.

Figure 4-1 Obtaining the addresses for connecting to Proxy Cluster DCS instances



The following example uses the domain name address of a DCS Redis instance. Change the domain name and port as required.

```
[root@ecs-redis redis-6.2.13]# cd src  
[root@ecs-redis src]# ./redis-cli -h redis-069949a-dcs-lxy.dcs.huaweicloud.com -p 6379  
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379>
```

- b. If you have set a password for the DCS instance, enter the password in this step. You can read and write cached data only after the password is verified. Skip this step if the instance is not password-protected.

auth {password}

{password} indicates the password used for logging in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

The command output is as follows:

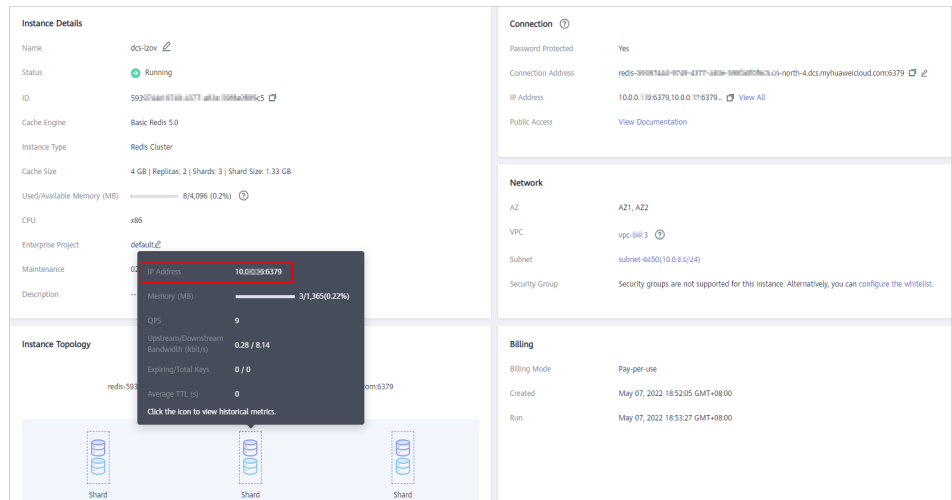
```
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379> auth *****  
OK  
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379>
```

- Access a DCS instance of the Redis Cluster type.
Perform the following procedure to access a Redis Cluster instance.
 - a. Run the following command to access the chosen DCS Redis instance:
`./redis-cli -h {dcs_instance_address} -p 6379 -a {password} -c`
{dcs_instance_address} indicates the IP address/domain name of the DCS Redis instance, **6379** is the port used for accessing the instance, *{password}* is the password of the instance, and **-c** is used for accessing Redis Cluster nodes. The IP address/domain name and port number are obtained in [Step 1](#).

NOTE

- You can set *{dcs_instance_address}* to **Connection Address** or **IP Address** in the **Connection** section, or **IP Address** in the **Instance Topology** section. The addresses can be obtained on the instance basic information page on the console, as shown in [Figure 4-2](#).
- If password-free access has been enabled for the instance, you do not need to enter the instance access password **-a {password}**. If you forget the password or need to reset the password, see [Resetting Instance Passwords](#).
- The **IP Address** field provides multiple IP addresses. You can use any of them to connect to the instance. The CRC16(key) mod 16384 algorithm is used to compute what is the hash slot of a given key. For higher reliability, configure all IP addresses.
- By using the **IP Address** in the **Instance Topology** section, you can connect to the specified shard.

Figure 4-2 Obtaining the addresses for connecting to a Redis Cluster DCS instance



- The following example uses the IP address of a DCS Redis instance. Change the IP address and port as required.

```
root@ecs-redis:~/redis-6.2.13/src# ./redis-cli -h 192.168.0.85 -p 6379 -a ***** -c 192.168.0.85:6379>
```
- The following example uses the domain name of a DCS Redis instance. Change the domain name and port as required.

```
root@ecs-redis:~/redis-6.2.13/src# ./redis-cli -h redis-51e463c-dcs-lxy.dcs.huaweicloud.com
-p 6379 -a ***** -c
redis-51e463c-dcs-lxy.dcs.huaweicloud.com:6379>
```

- b. Run the following command to view the Redis Cluster node information:

cluster nodes

Each shard in a Redis Cluster has a master and a replica by default. The proceeding command provides all the information of cluster nodes.

```
192.168.0.85:6379> cluster nodes
0988ae8fd3686074c9afdce73d7878c81a33ddc 192.168.0.231:6379@16379 slave
f0141816260ca5029c56333095f015c7a058f113 0 1568084030
000 3 connected
1a32d809c0b743bd83b5e1c277d5d201d0140b75 192.168.0.85:6379@16379 myself,master - 0
1568084030000 2 connected 5461-10922
c8ad7af9a12cce3c8e416fb67bd6ec9207f0082d 192.168.0.130:6379@16379 slave
1a32d809c0b743bd83b5e1c277d5d201d0140b75 0 1568084031
000 2 connected
7ca218299c254b5da939f8e60a940ac8171adc27 192.168.0.22:6379@16379 master - 0
1568084030000 1 connected 0-5460
f0141816260ca5029c56333095f015c7a058f113 192.168.0.170:6379@16379 master - 0
1568084031992 3 connected 10923-16383
19b1a400815396c6223963b013ec934a657bdc52 192.168.0.161:6379@16379 slave
7ca218299c254b5da939f8e60a940ac8171adc27 0 1568084031
000 1 connected
```

Write operations can only be performed on master nodes. The CRC16(key) mod 16384 algorithm is used to compute what is the hash slot of a given key.

As shown in the following, the value of **CRC16 (KEY) mode 16384** determines the hash slot that a given key is located at and redirects the client to the node where the hash slot is located at.

```
192.168.0.170:6379> set hello world
-> Redirected to slot [866] located at 192.168.0.22:6379
OK
192.168.0.22:6379> set happy day
OK
192.168.0.22:6379> set abc 123
-> Redirected to slot [7638] located at 192.168.0.85:6379
OK
192.168.0.85:6379> get hello
-> Redirected to slot [866] located at 192.168.0.22:6379
"world"
192.168.0.22:6379> get abc
-> Redirected to slot [7638] located at 192.168.0.85:6379
"123"
192.168.0.85:6379>
```

----End

Procedure (Windows)

Click [here](#) to download the Redis client installation package for Windows. Decompress the package in any directory, open the CLI tool **cmd.exe**, and go to the directory. Then, run the following command to access the DCS Redis instance:

```
redis-cli.exe -h XXX -p 6379
```

XXX indicates the IP address/domain name of the DCS instance and **6379** is an example port number used for accessing the DCS instance. For details about how to obtain the IP address/domain name and port number, see [Viewing Instance Details](#).

4.3 Access in Different Languages

4.3.1 Java

4.3.1.1 Jedis

Access a DCS Redis instance through Jedis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

Spring Data Redis is already integrated with [Jedis](#) and [Lettuce](#) for Spring Boot projects. Spring Boot 1.x is integrated with Jedis, and Spring Boot 2.x is integrated with Lettuce. To use Jedis in Spring Boot 2.x and later, you need to solve Lettuce dependency conflicts.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed.
For details, see [Viewing Instance Details](#).
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the Java compilation environment has been installed on the ECS.

Pom Configuration

```
<!-- import spring-data-redis -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
<!--In Spring Boot 2.0, Lettuce is used by default. To use Jedis, solve dependency conflicts.-->
  <exclusions>
    <exclusion>
      <groupId>io.lettuce</groupId>
      <artifactId>lettuce-core</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!--Jedis dependency>
<dependency>
  <groupId>redis.clients</groupId>
  <artifactId>jedis</artifactId>
  <version>3.10.0</version>
</dependency>
```

application.properties Configuration

- Single-node, master/standby, read/write splitting, and Proxy Cluster
#Redis host
spring.redis.host=<host>
#Redis port
spring.redis.port=<port>
#Redis database number
spring.redis.database=0

```
#Redis password
spring.redis.password=<password>
#Redis read/write timeout
spring.redis.timeout=2000
#Whether to enable connection pooling
spring.redis.jedis.pool.enabled=true
#Minimum connections in the pool
spring.redis.jedis.pool.min-idle=50
#Maximum idle connections in the pool
spring.redis.jedis.pool.max-idle=200
#Maximum connections in the pool
spring.redis.jedis.pool.max-active=200
#Maximum amount of time a connection allocation should block before throwing an exception when
the pool is exhausted. The default value -1 indicates to wait indefinitely.
spring.redis.jedis.pool.max-wait=3000
#Interval for checking and evicting idle connection. Default: 60s.
spring.redis.jedis.pool.time-between-eviction-runs=60S
```

- **Redis Cluster**

```
#Redis Cluster node connection information
spring.redis.cluster.nodes=<ip:port>,<ip:port>,<ip:port>
#Redis Cluster password
spring.redis.password=<password>
#Redis Cluster max. redirecting times
spring.redis.cluster.max-redirects=3
#Redis read/write timeout
spring.redis.timeout=2000
#Whether to enable connection pooling
spring.redis.jedis.pool.enabled=true
#Minimum connections in the pool
spring.redis.jedis.pool.min-idle=50
#Maximum idle connections in the pool
spring.redis.jedis.pool.max-idle=200
#Maximum connections in the pool
spring.redis.jedis.pool.max-active=200
#Maximum amount of time a connection allocation should block before throwing an exception when
the pool is exhausted. The default value -1 indicates to wait indefinitely.
spring.redis.jedis.pool.max-wait=3000
#Interval for checking and evicting idle connections. Default: 60s.
spring.redis.jedis.pool.time-between-eviction-runs=60S
```

Bean Configuration

- **Single-node, master/standby, read/write splitting, and Proxy Cluster**

```
import java.time.Duration;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;

import redis.clients.jedis.JedisPoolConfig;

@Configuration
public class RedisConfiguration {

    @Value("${redis.host}")
    private String redisHost;

    @Value("${redis.port:6379}")
    private Integer redisPort = 6379;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password:}")
```



```
private String redisPassword;

@Value("${redis.connect.timeout:3000}")
private Integer redisConnectTimeout = 3000;

@Value("${redis.read.timeout:2000}")
private Integer redisReadTimeout = 2000;

@Value("${redis.pool.minSize:50}")
private Integer redisPoolMinSize = 50;

@Value("${redis.pool.maxSize:200}")
private Integer redisPoolMaxSize = 200;

@Value("${redis.pool.maxWaitMillis:3000}")
private Integer redisPoolMaxWaitMillis = 3000;

@Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

@Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

@Bean
public RedisConnectionFactory redisConnectionFactory(JedisClientConfiguration
clientConfiguration) {

    RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
    standaloneConfiguration.setHostName(redisHost);
    standaloneConfiguration.setPort(redisPort);
    standaloneConfiguration.setDatabase(redisDatabase);
    standaloneConfiguration.setPassword(redisPassword);

    return new JedisConnectionFactory(standaloneConfiguration, clientConfiguration);
}

@Bean
public JedisClientConfiguration clientConfiguration() {

    JedisClientConfiguration clientConfiguration = JedisClientConfiguration.builder()
        .connectTimeout(Duration.ofMillis(redisConnectTimeout))
        .readTimeout(Duration.ofMillis(redisReadTimeout))
        .usePooling().poolConfig(redisPoolConfig())
        .build();

    return clientConfiguration;
}

private JedisPoolConfig redisPoolConfig() {

    JedisPoolConfig poolConfig = new JedisPoolConfig();
    //Minimum connections in the pool
    poolConfig.setMinIdle(redisPoolMinSize);
    //Maximum idle connections in the pool
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //Maximum total connections in the pool
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //Wait when pool is exhausted? Set to true to wait. To validate setMaxWait, it has to be true.
    poolConfig.setBlockWhenExhausted(true);
    //Longest time to wait for connection after pool is exhausted. The default value -1 indicates to
    wait indefinitely.
    poolConfig.setMaxWaitMillis(redisPoolMaxWaitMillis);
    //Set to true to enable connectivity test on creating connections. Default: false.
    poolConfig.setTestOnCreate(false);
    //Set to true to enable connectivity test on borrowing connections. Default: false. Set to false for
    heavy-traffic services to reduce overhead.
    poolConfig.setTestOnBorrow(true);
    //Set to true to enable connectivity test on returning connections. Default: false. Set to false for
    heavy-traffic services to reduce overhead.
```

```
poolConfig.setTestOnReturn(false);
//Indicates whether to check for idle connections. If this is set to false, idle connections are not
evicted.
poolConfig.setTestWhileIdle(true);
//Duration after which idle connections are evicted. If the idle duration is greater than this value
and the maximum number of idle connections is reached, idle connections are directly evicted.
poolConfig.setSoftMinEvictableIdleTimeMillis(redisPoolSoftMinEvictableIdleTimeMillis);
//Disable MinEvictableIdleTimeMillis().
poolConfig.setMinEvictableIdleTimeMillis(-1);
//Interval for checking and evicting idle connections. Default: 60s.
poolConfig.setTimeBetweenEvictionRunsMillis(redisPoolBetweenEvictionRunsMillis);
return poolConfig;
}
}
```

- **Redis Cluster**

```
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;

import redis.clients.jedis.JedisPoolConfig;

@Configuration
public class RedisConfiguration {

    @Value("${redis.cluster.nodes}")
    private String redisClusterNodes;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.pool.minSize:50}")
    private Integer redisPoolMinSize = 50;

    @Value("${redis.pool.maxSize:200}")
    private Integer redisPoolMaxSize = 200;

    @Value("${redis.pool.maxWaitMillis:3000}")
    private Integer redisPoolMaxWaitMillis = 3000;

    @Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
    private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

    @Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
    private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

    @Bean
    public RedisConnectionFactory redisConnectionFactory(JedisClientConfiguration
clientConfiguration) {

        RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

        List<RedisNode> clusterNodes = new ArrayList<>();
        for (String clusterNodeStr : redisClusterNodes.split(",")) {
            String[] nodeInfo = clusterNodeStr.split(":");
```

```
        clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
    }
    clusterConfiguration.setClusterNodes(clusterNodes);

    clusterConfiguration.setPassword(redisPassword);
    clusterConfiguration.setMaxRedirects(3);

    return new JedisConnectionFactory(clusterConfiguration, clientConfiguration);
}

@Bean
public JedisClientConfiguration clientConfiguration() {

    JedisClientConfiguration clientConfiguration = JedisClientConfiguration.builder()
        .connectTimeout(Duration.ofMillis(redisConnectTimeout))
        .readTimeout(Duration.ofMillis(redisReadTimeout))
        .usePooling().poolConfig(redisPoolConfig())
        .build();

    return clientConfiguration;
}

private JedisPoolConfig redisPoolConfig() {

    JedisPoolConfig poolConfig = new JedisPoolConfig();
    //Minimum connections in the pool
    poolConfig.setMinIdle(redisPoolMinSize);
    //Maximum idle connections in the pool
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //Maximum total connections in the pool
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //Wait when pool is exhausted? Set to true to wait. To validate setMaxWait, it has to be true.
    poolConfig.setBlockWhenExhausted(true);
    //Longest time to wait for connection after pool is exhausted. The default value -1 indicates to
wait indefinitely.
    poolConfig.setMaxWaitMillis(redisPoolMaxWaitMillis);
    //Set to true to enable connectivity test on creating connections. Default: false.
    poolConfig.setTestOnCreate(false);
    //Set to true to enable connectivity test on borrowing connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnBorrow(true);
    //Set to true to enable connectivity test on returning connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnReturn(false);
    //Indicates whether to check for idle connections. If this is set to false, idle connections are not
evicted.
    poolConfig.setTestWhileIdle(true);
    //Duration after which idle connections are evicted. If the idle duration is greater than this value
and the maximum number of idle connections is reached, idle connections are directly evicted.
    poolConfig.setSoftMinEvictableIdleTimeMillis(redisPoolSoftMinEvictableIdleTimeMillis);
    //Disable MinEvictableIdleTimeMillis().
    poolConfig.setMinEvictableIdleTimeMillis(-1);
    //Interval for checking and evicting idle connections. Default: 60s.
    poolConfig.setTimeBetweenEvictionRunsMillis(redisPoolBetweenEvictionRunsMillis);
    return poolConfig;
}
}
```

Parameter Description

Table 4-1 RedisStandaloneConfiguration parameters

Parameter	Default Value	Description
hostName	localhost	IP address/domain name for connecting to a DCS Redis instance
port	6379	Port number
database	0	Database number. Default: 0.
password	-	Password

Table 4-2 RedisClusterConfiguration parameters

Parameter	Description
clusterNodes	Cluster node connection information, including the node IP address and port number
maxRedirects	Maximum redirecting times
password	Password

Table 4-3 JedisPoolConfig parameters

Parameter	Default Value	Description
minIdle	-	Minimum connections in the connection pool
maxIdle	-	Maximum idle connections in the connection pool
maxTotal	-	Maximum total connections in the connection pool
blockWhenExhausted	true	Indicates whether to wait after the connection pool is exhausted. true : Wait. false : Do not wait. To validate maxWaitMillis , this parameter must be set to true .
maxWaitMillis	-1	Maximum amount of time (in milliseconds) to wait for connection after the connection pool is exhausted. The default value -1 indicates to wait indefinitely.

Parameter	Default Value	Description
testOnCreate	false	Indicates whether to enable connectivity test on creating connections. false : Disable. true : Enable.
testOnBorrow	false	Indicates whether to enable connectivity test on obtaining connections. false : Disable. true : Enable. For heavy-traffic services, set this parameter to false to reduce overhead.
testOnReturn	false	Indicates whether to enable connectivity test on returning connections. false : Disable. true : Enable. For heavy-traffic services, set this parameter to false to reduce overhead.
testWhileIdle	false	Indicates whether to check for idle connections. If this parameter is set to false , idle connections are not evicted. Recommended value: true .
softMinEvictableIdleTimeMillis	1800000	Duration (in milliseconds) after which idle connections are evicted. If the idle duration is greater than this value and the maximum number of idle connections is reached, idle connections are directly evicted.
minEvictableIdleTimeMillis	60000	Minimum amount of time (in milliseconds) a connection may remain idle in the pool before it is eligible for eviction. The recommended value is -1 , indicating that softMinEvictableIdleTimeMillis is used instead.
timeBetweenEvictionRunsMillis	60000	Interval (in milliseconds) for checking and evicting idle connections.

Table 4-4 JedisClientConfiguration parameters

Parameter	Default Value	Description
connectTimeout	2000	Connection timeout interval, in milliseconds.
readTimeout	2000	Timeout interval for waiting for a response, in milliseconds.
poolConfig	-	Pool configurations. For details, see JedisPoolConfig .

Suggestion for Configuring DCS Instances

- Connection pool configuration

NOTE

The following calculation is applicable only to common service scenarios. You can customize it based on your service requirements.

There is no standard connection pool size. You can configure one based on your service traffic. The following formulas are for reference:

- Minimum number of connections = (QPS of a single node accessing Redis)/(1000 ms/Average time spent on a single command)
- Maximum number of connections = (QPS of a single node accessing Redis)/(1000 ms/Average time spent on a single command) x 150%

For example, if the QPS of a service application is about 10,000, each request needs to access Redis 10 times (that is, 100,000 accesses to Redis every second), and the service application has 10 hosts, the calculation is as follows:

QPS of a single node accessing Redis = 100,000/10 = 10,000

Average time spent on a single command = 20 ms (Redis takes 5 ms to 10 ms to process a single command under normal conditions. If network jitter occurs, it takes 15 ms to 20 ms.)

Minimum number of connections = 10,000/(1000 ms/20 ms) = 200

Maximum number of connections = 10,000/(1000 ms/20 ms) x 150% = 300

4.3.1.2 Lettuce

Access a DCS Redis instance through Lettuce on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

Spring Data Redis is already integrated with [Jedis](#) and [Lettuce](#) for Spring Boot projects. In addition, Spring Boot 1.x is integrated with Jedis, and Spring Boot 2.x with Lettuce. Therefore, you do not need to import Lettuce in Spring Boot 2.x and later projects.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the Java compilation environment has been installed on the ECS.

Pom Configuration

```
<!-- Enable Spring Data Redis, Lettuce-supported SDK is integrated by default -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

application.properties Configuration

- Single-node, master/standby, read/write splitting, and Proxy Cluster

```
#Redis host
spring.redis.host=<host>
#Redis port
spring.redis.port=<port>
#Redis database number
spring.redis.database=0
#Redis password
spring.redis.password=<password>
#Redis read/write timeout
spring.redis.timeout=2000
```

- Redis Cluster

```
#Redis Cluster node information
spring.redis.cluster.nodes=<ip:port>,<ip:port>,<ip:port>
#Redis Cluster max redirecting times
spring.redis.cluster.max-redirects=3
#Redis Cluster node password
spring.redis.password=<password>
#Redis Cluster timeout
spring.redis.timeout=2000
#Enable adaptive topology refresh
spring.redis.lettuce.cluster.refresh.adaptive=true
#Enable topology refresh every 10 seconds
spring.redis.lettuce.cluster.refresh.period=10S
```

Bean Configuration

- Configuration for single-node, master/standby, read/write splitting, and Proxy Cluster instances

```
import java.time.Duration;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;

/**
 * Lettuce non-pooling configuration (use either this or the application.properties configuration)
 */
@Configuration
public class RedisConfiguration {

    @Value("${redis.host}")
    private String redisHost;

    @Value("${redis.port:6379}")
    private Integer redisPort = 6379;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password:}")
    private String redisPassword;

    @Value("${redis.connect.timeout:2000}")
    private Integer redisConnectTimeout = 2000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;
```

```
@Bean
public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

    RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
    standaloneConfiguration.setHostName(redisHost);
    standaloneConfiguration.setPort(redisPort);
    standaloneConfiguration.setDatabase(redisDatabase);
    standaloneConfiguration.setPassword(redisPassword);

    LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(standaloneConfiguration, clientConfiguration);
    connectionFactory.setDatabase(redisDatabase);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClientOptions clientOptions = ClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .build();

    LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .clientOptions(clientOptions)
        .build();
    return clientConfiguration;
}
}
```

- Pooling configuration for single-node, master/standby, read/write splitting, and Proxy Cluster instances

Enable the pooling component

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-pool2</artifactId>
  <version>2.11.1</version>
</dependency>
```

Code

```
import java.time.Duration;

import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.connection.lettuce.LettucePoolingClientConfiguration;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;

/**
 * Lettuce pooling configuration
 */
@Configuration
public class RedisPoolConfiguration {
    @Value("${redis.host}")
```



```
private String redisHost;

@Value("${redis.port:6379}")
private Integer redisPort = 6379;

@Value("${redis.database:0}")
private Integer redisDatabase = 0;

@Value("${redis.password:}")
private String redisPassword;

@Value("${redis.connect.timeout:2000}")
private Integer redisConnectTimeout = 2000;

@Value("${redis.read.timeout:2000}")
private Integer redisReadTimeout = 2000;

@Value("${redis.pool.minSize:50}")
private Integer redisPoolMinSize = 50;

@Value("${redis.pool.maxSize:200}")
private Integer redisPoolMaxSize = 200;

@Value("${redis.pool.maxWaitMillis:2000}")
private Integer redisPoolMaxWaitMillis = 2000;

@Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

@Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

@Bean
public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

    RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
    standaloneConfiguration.setHostName(redisHost);
    standaloneConfiguration.setPort(redisPort);
    standaloneConfiguration.setDatabase(redisDatabase);
    standaloneConfiguration.setPassword(redisPassword);

    LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(standaloneConfiguration, clientConfiguration);
    connectionFactory.setDatabase(redisDatabase);
    //Disable sharing native connection before enabling pooling
    connectionFactory.setShareNativeConnection(false);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClientOptions clientOptions = ClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .build();

    LettucePoolingClientConfiguration poolingClientConfiguration =
LettucePoolingClientConfiguration.builder()
        .poolConfig(redisPoolConfig())
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .clientOptions(clientOptions)
```

```
        .build();
    return poolingClientConfiguration;
}

private GenericObjectPoolConfig redisPoolConfig() {
    GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
    //Minimum idle connections in the pool
    poolConfig.setMinIdle(redisPoolMinSize);
    //Maximum idle connections in the pool
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //Maximum total connections in the pool
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //Wait when pool is exhausted? Set to true to wait. To validate setMaxWait, it has to be true.
    poolConfig.setBlockWhenExhausted(true);
    //Max allowed time to wait for connection after pool is exhausted. The default value -1 indicates
to wait indefinitely.
    poolConfig.setMaxWait(Duration.ofMillis(redisPoolMaxWaitMillis));
    //Set to true to enable connectivity test on creating connections. Default: false.
    poolConfig.setTestOnCreate(false);
    //Set to true to enable connectivity test on borrowing connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnBorrow(true);
    //Set to true to enable connectivity test on returning connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnReturn(false);
    //Indicates whether to check for idle connections. If this is set to false, idle connections are not
evicted.
    poolConfig.setTestWhileIdle(true);
    //Idle duration after which a connection is evicted. If the actual duration is greater than this
value and the maximum number of idle connections is reached, idle connections are directly evicted.

    poolConfig.setSoftMinEvictableIdleTime(Duration.ofMillis(redisPoolSoftMinEvictableIdleTimeMillis));
    //Disable eviction policy MinEvictableIdleTimeMillis().
    poolConfig.setMinEvictableIdleTime(Duration.ofMillis(-1));
    //Interval for checking and evicting idle connections. Default: 60s.
    poolConfig.setTimeBetweenEvictionRuns(Duration.ofMillis(redisPoolBetweenEvictionRunsMillis));
    return poolConfig;
}
}
```

- Configuration for Redis Cluster instances

```
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;
import io.lettuce.core.cluster.ClusterClientOptions;
import io.lettuce.core.cluster.ClusterTopologyRefreshOptions;

/**
 * Lettuce Cluster non-pooling configuration (use either this or the application.properties configuration)
 */
@Configuration
public class RedisConfiguration {

    @Value("${redis.cluster.nodes}")
    private String redisClusterNodes;

    @Value("${redis.cluster.maxDirects:3}")
    private Integer redisClusterMaxDirects;
```

```
@Value("${redis.password}")
private String redisPassword;

@Value("${redis.connect.timeout:2000}")
private Integer redisConnectTimeout = 2000;

@Value("${redis.read.timeout:2000}")
private Integer redisReadTimeout = 2000;

@Value("${redis.cluster.topology.refresh.period.millis:10000}")
private Integer redisClusterTopologyRefreshPeriodMillis = 10000;

@Bean
public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

    RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

    List<RedisNode> clusterNodes = new ArrayList<>();
    for (String clusterNodeStr : redisClusterNodes.split(",")) {
        String[] nodeInfo = clusterNodeStr.split(":");
        clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
    }
    clusterConfiguration.setClusterNodes(clusterNodes);

    clusterConfiguration.setPassword(redisPassword);
    clusterConfiguration.setMaxRedirects(redisClusterMaxDirects);

    LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(clusterConfiguration, clientConfiguration);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
        .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
        .build();

    ClusterClientOptions clientOptions = ClusterClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();

    LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .clientOptions(clientOptions)
        .build();
    return clientConfiguration;
}
}
```

- Pooling configuration for Redis Cluster instances

Enable the pooling component

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

```
<version>2.11.1</version>
</dependency>

Code

import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.connection.lettuce.LettucePoolingClientConfiguration;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;
import io.lettuce.core.cluster.ClusterClientOptions;
import io.lettuce.core.cluster.ClusterTopologyRefreshOptions;

/**
 * Lettuce pooling configuration
 */
@Configuration
public class RedisPoolConfiguration {

    @Value("${redis.cluster.nodes}")
    private String redisClusterNodes;

    @Value("${redis.cluster.maxDirects:3}")
    private Integer redisClusterMaxDirects;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:2000}")
    private Integer redisConnectTimeout = 2000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.cluster.topology.refresh.period.millis:10000}")
    private Integer redisClusterTopologyRefreshPeriodMillis = 10000;

    @Value("${redis.pool.minSize:50}")
    private Integer redisPoolMinSize = 50;

    @Value("${redis.pool.maxSize:200}")
    private Integer redisPoolMaxSize = 200;

    @Value("${redis.pool.maxWaitMillis:2000}")
    private Integer redisPoolMaxWaitMillis = 2000;

    @Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
    private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

    @Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
    private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

    @Bean
    public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

        RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

        List<RedisNode> clusterNodes = new ArrayList<>();
```

```
for (String clusterNodeStr : redisClusterNodes.split(",")) {
    String[] nodeInfo = clusterNodeStr.split(":");
    clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
}
clusterConfiguration.setClusterNodes(clusterNodes);

clusterConfiguration.setPassword(redisPassword);
clusterConfiguration.setMaxRedirects(redisClusterMaxDirects);

LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(clusterConfiguration, clientConfiguration);
//Disable native connection sharing before validating connection pool
connectionFactory.setShareNativeConnection(false);
return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
        .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
        .build();

    ClusterClientOptions clientOptions = ClusterClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();

    LettucePoolingClientConfiguration clientConfiguration =
LettucePoolingClientConfiguration.builder()
        .poolConfig(poolConfig())
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .clientOptions(clientOptions)
        .build();
    return clientConfiguration;
}

private GenericObjectPoolConfig poolConfig() {
    GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
    //Minimum connections in the pool
    poolConfig.setMinIdle(redisPoolMinSize);
    //Maximum idle connections in the pool
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //Maximum total connections in the pool
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //Wait when pool is exhausted? Set to true to wait. To validate setMaxWait, it has to be true.
    poolConfig.setBlockWhenExhausted(true);
    //Max allowed time to wait for connection after pool is exhausted. The default value -1 indicates
to wait indefinitely.
    poolConfig.setMaxWait(Duration.ofMillis(redisPoolMaxWaitMillis));
    //Set to true to enable connectivity test on creating connections. Default: false.
    poolConfig.setTestOnCreate(false);
    //Set to true to enable connectivity test on borrowing connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnBorrow(true);
    //Set to true to enable connectivity test on returning connections. Default: false. Set to false for
heavy-traffic services to reduce overhead.
    poolConfig.setTestOnReturn(false);
    //Indicates whether to check for idle connections. If this is set to false, idle connections are not
evicted.
}
```

```

poolConfig.setTestWhileIdle(true);
//Disable connection closure when the minimum idle time is reached.
poolConfig.setMinEvictableIdleTime(Duration.ofMillis(-1));
//Idle duration before a connection being evicted. If the actual duration is greater than this value
and the maximum number of idle connections is reached, idle connections are directly evicted.
MinEvictableIdleTimeMillis (default eviction policy) is no longer used.

poolConfig.setSoftMinEvictableIdleTime(Duration.ofMillis(redisPoolSoftMinEvictableIdleTimeMillis));
//Interval for checking and evicting idle connections. Default: 60s.
poolConfig.setTimeBetweenEvictionRuns(Duration.ofMillis(redisPoolBetweenEvictionRunsMillis));

return poolConfig;
}
}

```

Parameter Description

Table 4-5 LettuceConnectionFactory parameters

Parameter	Type	Default Value	Description
configuration	RedisConfiguration	-	Redis connection configuration. Two subclasses: <ul style="list-style-type: none"> RedisStandaloneConfiguration RedisClusterConfiguration
clientConfiguration	LettuceClientConfiguration	-	Client configuration parameter. Common subclass: LettucePoolingClientConfiguration
shareNativeConnection	boolean	true	Indicates whether to share native connections. Set to true to share. Set to false to enable connection pooling.

Table 4-6 RedisStandaloneConfiguration parameters

Parameter	Default Value	Description
hostName	localhost	IP address/domain name for connecting to a DCS Redis instance
port	6379	Port number
database	0	Database subscript
password	-	Password

Table 4-7 RedisClusterConfiguration parameters

Parameter	Description
clusterNodes	Cluster node connection information, including the node IP address and port number
maxRedirects	Maximum redirecting times. Recommended value: 3 .
password	Password

Table 4-8 LettuceClientConfiguration parameters

Parameter	Type	Default Value	Description
timeout	Duration	60s	Command timeout: Recommended: 2s .
clientOptions	ClientOptions	-	Configuration options.

Table 4-9 LettucePoolingClientConfiguration parameters

Parameter	Type	Default Value	Description
timeout	Duration	60s	Command timeout: Recommended: 2s .
clientOptions	ClientOptions	-	Configuration options.
poolConfig	GenericObjectPoolConfig	-	Connection pool configuration.

Table 4-10 ClientOptions parameters

Parameter	Type	Default Value	Description
autoReconnect	boolean	true	Indicates whether to automatically reconnect after disconnection. Recommended: true .
pingBeforeActivateConnection	boolean	true	Indicates whether to test connectivity on established connections. Recommended: true .

Parameter	Type	Default Value	Description
cancelCommandsOnReconnectFailure	boolean	true	Indicates whether to cancel commands after a failed reconnection attempt. Recommended: false .
disconnectedBehavior	DisconnectedBehavior	DisconnectedBehavior.DEFAULT	Indicates what to do when a connection drops. Recommended: ACCEPT_COMMANDS . <ul style="list-style-type: none"> • DEFAULT: When autoReconnect is set true, commands are allowed to wait in queue. When autoReconnect is set to false, commands are not allowed to wait in queue. • ACCEPT_COMMANDS: Allow commands to wait in queue. • REJECT_COMMANDS: Do not allow commands to wait in queue.
socketOptions	SocketOptions	-	Socket configuration.

Table 4-11 SocketOptions parameters

Parameter	Default Value	Description
connectTimeout	10s	Connection timeout. Recommended: 2s .

Table 4-12 GenericObjectPoolConfig parameters

Parameter	Default Value	Description
minIdle	-	Minimum connections in the pool.
maxIdle	-	Maximum idle connections in the connection pool.
maxTotal	-	Maximum total connections in the connection pool.

Parameter	Default Value	Description
blockWhenExhausted	true	Indicates whether to wait after the connection pool is exhausted. true : Wait. false : Do not wait. To validate maxWaitMillis , this parameter must be set to true .
maxWaitMillis	-1	Maximum amount of time a connection allocation should block before throwing an exception when the pool is exhausted. The default value -1 indicates to wait indefinitely.
testOnCreate	false	Set to true to enable connectivity test on creating connections. Default: false .
testOnBorrow	false	Set to true to enable connectivity test on borrowing connections. Default: false . Set to false for heavy-traffic services to reduce overhead.
testOnReturn	false	Set to true to enable connectivity test on returning connections. Default: false . Set to false for heavy-traffic services to reduce overhead.
testWhileIdle	false	Indicates whether to check for idle connections. If this parameter is set to false , idle connections are not evicted. Recommended value: true .
softMinEvictableIdleTimeMillis	1800000	Duration after which idle connections are evicted. If the idle duration is greater than this value and the maximum number of idle connections is reached, idle connections are directly evicted.
minEvictableIdleTimeMillis	60000	An eviction policy, set to -1 (suggested) to disable it. Use softminEvictableIdleTimeMillis instead.
timeBetweenEvictionRunsMillis	60000	Eviction interval, in milliseconds.

Suggestion for Configuring DCS Instances

- Pooling connection

Different from Jedis's BIO, the bottom layer of Lettuce communicates with Redis Server based on Netty's NIO. Combining persistent connections and queues, Lettuce sends and receives multiple requests and responses spontaneously with sequential sending and receiving features of TCP. A single connection supports 3000 to 5000 QPS, but you are not advised to allow more than 3000 QPS in production systems. Pooling is not supported by Lettuce,

and is disabled by default in Spring Boot. To enable pooling, validate the commons-pool2 dependency and disable native connection sharing.

By default, each Lettuce connection needs two thread pools, I/O thread pool and computation thread pool, to support I/O event reading and asynchronous event processing. If you configure connection pooling, each connection creates two thread pools, consuming high memory resources. **Lettuce is strong at processing single connections based on its bottom-layer implementation, so you are not advised to use Lettuce with pooling.**

- Topology refresh

When connecting to a Redis Cluster instance, Lettuce randomly sends **cluster nodes** to the node list during initialization to obtain the distribution of cluster slots. Cluster topology structure changes when the cluster capacity is increased or decreased or a master/standby switchover occurs. Lettuce does not detect such changes by default. You can enable detection with the following configurations:

- **application.properties configuration**

```
#Enable adaptive topology refresh.
spring.redis.lettuce.cluster.refresh.adaptive=true
#Enable topology refresh every 10 seconds.
spring.redis.lettuce.cluster.refresh.period=10S
```

- **API configuration**

```
ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
    .build();

ClusterClientOptions clientOptions = ClusterClientOptions.builder()
    ...
    ...
    .topologyRefreshOptions(topologyRefreshOptions)
    .build();
```

- Blast radius

The bottom layer of Lettuce uses a combination of single persistent connection and request queue. Once network jitter or intermittent disconnection occurs or connection times out, all requests are affected. Especially when connection times out, an attempt is made to resend TCP packets until timeout and connection drops. Requests do not recover until connections are reestablished. Requests accumulate during resending attempts. If upper-layer services time out in batches, or the resending timeout is too long in some OS' kernels, the service system remains unavailable for a long time. **Therefore, you are advised to use Jedis instead of Lettuce.**

4.3.1.3 Redisson

Access a DCS Redis instance through Redisson on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

For Spring Boot projects, Spring Data Redis is already integrated with [Jedis](#) and [Lettuce](#), but does not support Redisson. [Redisson](#) provides the redisson-spring-boot-starter component (<https://mvnrepository.com/artifact/org.redisson/redisson>) that can be used with Spring Boot.

Spring Boot 1.x is integrated with Jedis, and Spring Boot 2.x is integrated with Lettuce.

 NOTE

- If a password was set during DCS Redis instance creation, configure the password for connecting to Redis using Redisson. Do not hard code the plaintext password.
- To connect to a single-node, read/write splitting, or Proxy Cluster instance, use the **useSingleServer** method of the **SingleServerConfig** object of Redisson. To connect to a master/standby instance, use the **useMasterSlaveServers** method of the **MasterSlaveServersConfig** object of Redisson. To connect to a Redis Cluster instance, use the **useClusterServers** method of the **ClusterServersConfig** object.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the Java compilation environment has been installed on the ECS.

Pom Configuration

```
<!-- spring-data-redis -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
  <exclusions>
    <!--Lettuce is integrated in Spring Boot 2.x by default. This dependency needs to be deleted. -->
    <exclusion>
      <artifactId>lettuce-core</artifactId>
      <groupId>io.lettuce</groupId>
    </exclusion>
  </exclusions>
</dependency>
<!--Redisson's adaptation package for Spring Boot-->
<dependency>
  <groupId>org.redisson</groupId>
  <artifactId>redisson-spring-boot-starter</artifactId>
  <version>${redisson.version}</version>
</dependency>
```

Bean Configuration

Spring Boot does not provide Redisson adaptation, and the **application.properties** configuration file does not have the corresponding configuration item. Therefore, you can only use Bean configuration.

- Single-node, read/write splitting, and Proxy Cluster

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.Config;
import org.redisson.config.SingleServerConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SingleConfig {
```

```
@Value("${redis.address}")
private String redisAddress;

@Value("${redis.password}")
private String redisPassword;

@Value("${redis.database:0}")
private Integer redisDatabase = 0;

@Value("${redis.connect.timeout:3000}")
private Integer redisConnectTimeout = 3000;

@Value("${redis.connection.idle.timeout:10000}")
private Integer redisConnectionIdleTimeout = 10000;

@Value("${redis.connection.ping.interval:1000}")
private Integer redisConnectionPingInterval = 1000;

@Value("${redis.timeout:2000}")
private Integer timeout = 2000;

@Value("${redis.connection.pool.min.size:50}")
private Integer redisConnectionPoolMinSize;

@Value("${redis.connection.pool.max.size:200}")
private Integer redisConnectionPoolMaxSize;

@Value("${redis.retry.attempts:3}")
private Integer redisRetryAttempts = 3;

@Value("${redis.retry.interval:200}")
private Integer redisRetryInterval = 200;

@Bean
public RedissonClient redissonClient(){
    Config redissonConfig = new Config();

    SingleServerConfig serverConfig = redissonConfig.useSingleServer();
    serverConfig.setAddress(redisAddress);
    serverConfig.setConnectionMinimumIdleSize(redisConnectionPoolMinSize);
    serverConfig.setConnectionPoolSize(redisConnectionPoolMaxSize);

    serverConfig.setDatabase(redisDatabase);
    serverConfig.setPassword(redisPassword);
    serverConfig.setConnectTimeout(redisConnectTimeout);
    serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
    serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
    serverConfig.setTimeout(timeout);
    serverConfig.setRetryAttempts(redisRetryAttempts);
    serverConfig.setRetryInterval(redisRetryInterval);

    redissonConfig.setCodec(new JsonJacksonCodec());
    return Redisson.create(redissonConfig);
}
}
```

- **Master/Standby**

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.Config;
import org.redisson.config.MasterSlaveServersConfig;
import org.redisson.config.ReadMode;
import org.redisson.config.SubscriptionMode;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.HashSet;
```

```
@Configuration
public class MasterStandbyConfig {
    @Value("${redis.master.address}")
    private String redisMasterAddress;

    @Value("${redis.slave.address}")
    private String redisSlaveAddress;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password:}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.connection.idle.timeout:10000}")
    private Integer redisConnectionIdleTimeout = 10000;

    @Value("${redis.connection.ping.interval:1000}")
    private Integer redisConnectionPingInterval = 1000;

    @Value("${redis.timeout:2000}")
    private Integer timeout = 2000;

    @Value("${redis.master.connection.pool.min.size:50}")
    private Integer redisMasterConnectionPoolMinSize = 50;

    @Value("${redis.master.connection.pool.max.size:200}")
    private Integer redisMasterConnectionPoolMaxSize = 200;

    @Value("${redis.retry.attempts:3}")
    private Integer redisRetryAttempts = 3;

    @Value("${redis.retry.interval:200}")
    private Integer redisRetryInterval = 200;

    @Bean
    public RedissonClient redissonClient() {
        Config redissonConfig = new Config();

        MasterSlaveServersConfig serverConfig = redissonConfig.useMasterSlaveServers();
        serverConfig.setMasterAddress(redisMasterAddress);
        HashSet<String> slaveSet = new HashSet<>();
        slaveSet.add(redisSlaveAddress);
        serverConfig.setSlaveAddresses(slaveSet);

        serverConfig.setDatabase(redisDatabase);
        serverConfig.setPassword(redisPassword);

        serverConfig.setMasterConnectionMinimumIdleSize(redisMasterConnectionPoolMinSize);
        serverConfig.setMasterConnectionPoolSize(redisMasterConnectionPoolMaxSize);

        serverConfig.setReadMode(ReadMode.MASTER_SLAVE);
        serverConfig.setSubscriptionMode(SubscriptionMode.MASTER);

        serverConfig.setConnectTimeout(redisConnectTimeout);
        serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
        serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
        serverConfig.setTimeout(timeout);
        serverConfig.setRetryAttempts(redisRetryAttempts);
        serverConfig.setRetryInterval(redisRetryInterval);

        redissonConfig.setCodec(new JsonJacksonCodec());
        return Redisson.create(redissonConfig);
    }
}
```

- Redis Cluster

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.ClusterServersConfig;
import org.redisson.config.Config;
import org.redisson.config.ReadMode;
import org.redisson.config.SubscriptionMode;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.List;

@Configuration
public class ClusterConfig {

    @Value("${redis.cluster.address}")
    private List<String> redisClusterAddress;

    @Value("${redis.cluster.scan.interval:5000}")
    private Integer redisClusterScanInterval = 5000;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.connection.idle.timeout:10000}")
    private Integer redisConnectionIdleTimeout = 10000;

    @Value("${redis.connection.ping.interval:1000}")
    private Integer redisConnectionPingInterval = 1000;

    @Value("${redis.timeout:2000}")
    private Integer timeout = 2000;

    @Value("${redis.retry.attempts:3}")
    private Integer redisRetryAttempts = 3;

    @Value("${redis.retry.interval:200}")
    private Integer redisRetryInterval = 200;

    @Value("${redis.master.connection.pool.min.size:50}")
    private Integer redisMasterConnectionPoolMinSize = 50;

    @Value("${redis.master.connection.pool.max.size:200}")
    private Integer redisMasterConnectionPoolMaxSize = 200;

    @Bean
    public RedissonClient redissonClient() {
        Config redissonConfig = new Config();

        ClusterServersConfig serverConfig = redissonConfig.useClusterServers();
        serverConfig.setNodeAddresses(redisClusterAddress);
        serverConfig.setScanInterval(redisClusterScanInterval);

        serverConfig.setPassword(redisPassword);

        serverConfig.setMasterConnectionMinimumIdleSize(redisMasterConnectionPoolMinSize);
        serverConfig.setMasterConnectionPoolSize(redisMasterConnectionPoolMaxSize);

        serverConfig.setReadMode(ReadMode.MASTER);
        serverConfig.setSubscriptionMode(SubsriptionMode.MASTER);

        serverConfig.setConnectTimeout(redisConnectTimeout);
        serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
        serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
        serverConfig.setTimeout(timeout);
    }
}
```

```
serverConfig.setRetryAttempts(redisRetryAttempts);
serverConfig.setRetryInterval(redisRetryInterval);

redissonConfig.setCodec(new JsonJacksonCodec());
return Redisson.create(redissonConfig);
}
}
```

Parameter Description

Table 4-13 Config parameters

Parameter	Default Value	Description
codec	org.redisson.codec.JsonJacksonCodec	Encoding format, including JSON, Avro, Smile, CBOR, and MsgPack.
threads	Number of CPU cores x 2	Thread pool used for executing RTopic Listener, RRemoteService, and RExecutorService.
executor	null	The function is the same as threads . If this parameter is not set, a thread pool is initialized based on threads .
nettyThreads	Number of CPU cores x 2	Thread pool used by the TCP channel that connects to the redis-server. All channels share this connection pool and are mapped to Netty's Bootstrap.group(...) .
eventLoopGroup	null	The function is the same as nettyThreads . If this parameter is not set, an EventLoopGroup is initialized based on the nettyThreads parameter for the bottom-layer TCP channel to use.
transportMode	TransportMode.NIO	Transmission mode. The options are NIO , EPOLL (additional package required), and KQUEUE (additional package required).
lockWatchdogTimeout	30000	Timeout interval (in milliseconds) of the lock-monitoring watchdog. In the distributed lock scenario, if the leaseTimeout parameter is not specified, the default value of this parameter is used.
keepPubSubOrder	true	Indicates whether to receive messages in the publish sequence. If messages can be processed concurrently, you are advised to set this parameter to false.

Table 4-14 SingleServerConfig parameters (single-node, read/write splitting,, or Proxy Cluster)

Parameter	Default Value	Description
address	-	Node connection information, in redis:// <i>ip:port</i> format.
database	0	ID of the database to be used.
connectionMinimumIdleSize	32	Minimum number of connections to the master node of each shard.
connectionPoolSize	64	Maximum number of connections to the master node of each shard.
subscriptionConnectionMinimumIdleSize	1	Minimum number of connections to the target node for pub/sub.
subscriptionConnectionPoolSize	50	Maximum number of connections to the target node for pub/sub.
subscriptionPerConnection	5	Maximum number of subscriptions on each subscription connection.
connectionTimeout	10000	Connection timeout interval, in milliseconds.
idleConnectionTimeout	10000	Maximum time (in milliseconds) for reclaiming idle connections.
pingConnectionInterval	30000	Heartbeat for detecting available connections, in milliseconds. Recommended: 3000 ms.
timeout	3000	Timeout interval for waiting for a response, in milliseconds.
retryAttempts	3	Maximum number of retries upon send failures.
retryInterval	1500	Retry interval, in milliseconds. Recommended: 200 ms.
clientName	null	Client name.

Table 4-15 MasterSlaveServersConfig parameters (master/standby)

Parameter	Default Value	Description
masterAddress	-	Master node connection information, in redis:// <i>ip:port</i> format.
slaveAddresses	-	Standby node connection information, in Set<redis:// <i>ip:port</i> > format.

Parameter	Default Value	Description
readMode	SLAVE	Read mode. By default, read traffic is distributed to replica nodes. The value can be MASTER (recommended), SLAVE , or MASTER_SLAVE .
loadBalancer	RoundRobinLoad Balancer	Load balancing algorithm. This parameter is valid only when readMode is set to SLAVE or MASTER_SLAVE . Read traffic is distributed evenly.
masterConnectionMinimumIdleSize	32	Minimum number of connections to the master node of each shard.
masterConnectionPoolSize	64	Maximum number of connections to the master node of each shard.
slaveConnectionMinimumIdleSize	32	Minimum number of connections to each replica node of each shard. If readMode is set to MASTER , the value of this parameter is invalid.
slaveConnectionPoolSize	64	Maximum number of connections to each replica node of each shard. If readMode is set to MASTER , the value of this parameter is invalid.
subscriptionMode	SLAVE	Subscription mode. By default, only replica nodes handle subscription. The value can be SLAVE or MASTER (recommended).
subscriptionConnectionMinimumIdleSize	1	Minimum number of connections to the target node for pub/sub.
subscriptionConnectionPoolSize	50	Maximum number of connections to the target node for pub/sub.
subscriptionPerConnection	5	Maximum number of subscriptions on each subscription connection.
connectionTimeout	10000	Connection timeout interval, in milliseconds.
idleConnectionTimeout	10000	Maximum time (in milliseconds) for reclaiming idle connections.
pingConnectionInterval	30000	Heartbeat for detecting available connections, in milliseconds. Recommended: 3000 ms.
timeout	3000	Timeout interval for waiting for a response, in milliseconds.

Parameter	Default Value	Description
retryAttempts	3	Maximum number of retries upon send failures.
retryInterval	1500	Retry interval, in milliseconds. Recommended: 200 ms.
clientName	null	Client name.

Table 4-16 ClusterServersConfig parameters (Redis Cluster)

Parameter	Default Value	Description
nodeAddress	-	Connection addresses of cluster nodes. Each address uses the <code>redis://ip:port</code> format. Use commas (,) to separate connection addresses of different nodes.
password	null	Password for logging in to the cluster.
scanInterval	1000	Interval for periodically checking the cluster node status, in milliseconds.
readMode	SLAVE	Read mode. By default, read traffic is distributed to replica nodes. The value can be MASTER (recommended), SLAVE , or MASTER_SLAVE .
loadBalancer	RoundRobinLoadBalancer	Load balancing algorithm. This parameter is valid only when readMode is set to SLAVE or MASTER_SLAVE . Read traffic is distributed evenly.
masterConnectionMinimumIdleSize	32	Minimum number of connections to the master node of each shard.
masterConnectionPoolSize	64	Maximum number of connections to the master node of each shard.
slaveConnectionMinimumIdleSize	32	Minimum number of connections to each replica node of each shard. If readMode is set to MASTER , the value of this parameter is invalid.
slaveConnectionPoolSize	64	Maximum number of connections to each replica node of each shard. If readMode is set to MASTER , the value of this parameter is invalid.

Parameter	Default Value	Description
subscriptionMode	SLAVE	Subscription mode. By default, only replica nodes handle subscription. The value can be SLAVE or MASTER (recommended).
subscriptionConnectionMinimumIdleSize	1	Minimum number of connections to the target node for pub/sub.
subscriptionConnectionPoolSize	50	Maximum number of connections to the target node for pub/sub.
subscriptionPerConnection	5	Maximum number of subscriptions on each subscription connection.
connectionTimeout	10000	Connection timeout interval, in milliseconds.
idleConnectionTimeout	10000	Maximum time (in milliseconds) for reclaiming idle connections.
pingConnectionInterval	30000	Heartbeat for detecting available connections, in milliseconds. Recommended: 3000.
timeout	3000	Timeout interval for waiting for a response, in milliseconds.
retryAttempts	3	Maximum number of retries upon send failures.
retryInterval	1500	Retry interval, in milliseconds. Recommended: 200.
clientName	null	Client name.

Suggestion for Configuring DCS Instances

- **readMode**

MASTER is the recommended value, that is, the master node bears all read and write traffic. This is to avoid data inconsistency caused by master/replica synchronization latency. If the value is **SLAVE**, all read requests will trigger errors when replicas are faulty. If the value is **MASTER_SLAVE**, some read requests will trigger errors. Read errors last for the period specified by **failedSlaveCheckInterval** (180s by default) until the faulty nodes are removed from the available node list.

If read traffic and write traffic need to be separated, you can use read/write splitting DCS instances. Proxy nodes are deployed in the middle to distribute read and write traffic. When a replica node is faulty, traffic is automatically switched to the master node. The switchover does not interrupt service applications, and the fault detection time window is far shorter than Redisson's window.

- **subscriptionMode**
Similarly, **MASTER** is the recommended value.
- Connection pool configuration

 **NOTE**

The following calculation is applicable only to common service scenarios. You can customize it based on your service requirements.

There is no standard connection pool size. You can configure one based on your service traffic. The following formulas are for reference:

- Minimum number of connections = (QPS of a single node accessing Redis)/(1000 ms/Average time spent on a single command)
- Maximum number of connections = (QPS of a single node accessing Redis)/(1000 ms/Average time spent on a single command) x 150%

For example, if the QPS of a service application is about 10,000, each request needs to access Redis 10 times (that is, 100,000 accesses to Redis every second), and the service application has 10 hosts, the calculation is as follows:

QPS of a single node accessing Redis = 100,000/10 = 10,000

Average time spent on a single command = 20 ms (Redis takes 5 ms to 10 ms to process a single command under normal conditions. If network jitter occurs, it takes 15 ms to 20 ms.)

Minimum number of connections = 10,000/(1000 ms/20 ms) = 200

Maximum number of connections = 10,000/(1000 ms/20 ms) x 150% = 300

- Retry configuration
Redisson supports retries. You can set the following parameters based on service requirements. Generally, configure three retries, and set the retry interval to about 200 ms.
 - **retryAttempts**: number of retry times
 - **retryInterval**: retry interval

 **NOTE**

In Redisson, some APIs are implemented through LUA, and the performance is low. You are advised to use Jedis instead of Redisson.

4.3.2 Clients in Python

Access a DCS Redis instance through redis-py on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

 **NOTE**

Use redis-py to connect to single-node, master/standby, and Proxy Cluster instances and redis-py-cluster to connect to Redis Cluster instances.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).

- If the ECS runs the Linux OS, ensure that the Python compilation environment has been installed on the ECS.

Procedure

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

The following uses CentOS as an example to describe how to access an instance using a Python client.

Step 3 Access the DCS Redis instance.

If the system does not provide Python, run the following **yum** command to install it:

yum install python

NOTE

The Python version must be 3.6 or later. If the default Python version is earlier than 3.6, perform the following operations to change it:

1. Run the **rm -rf python** command to delete the Python symbolic link.
 2. Run the **ln -s python.X.X.X python** command to create another Python link. In the command, *X.X.X* indicates the Python version number.
- If the instance is a single-node, master/standby, or Proxy Cluster instance:
 - a. Install Python and redis-py.
 - i. If the system does not provide Python, run the **yum** command to install it.
 - ii. Run the following command to download and decompress the redis-py package:
wget https://github.com/andymccurdy/redis-py/archive/master.zip
unzip master.zip
 - iii. Go to the directory where the decompressed redis-py package is saved, and install redis-py.

python setup.py install

After the installation, run the **python** command. redis-py have been successfully installed if the following command output is displayed:

Figure 4-3 Running the python command

```
[root@ecs-2-1-1222223 redis-py-master]# python
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

- b. Use the redis-py client to connect to the instance. In the following steps, commands are executed in CLI mode. (Alternatively, write the commands into a Python script and then execute the script.)

- i. Run the **python** command to enter the CLI mode. You have entered CLI mode if the following command output is displayed:

Figure 4-4 Entering the CLI mode

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

- ii. Run the following command to access the chosen DCS Redis instance:

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

`XXX.XXX.XXX.XXX` indicates the IP address/domain name of the DCS instance and **6379** is an example port number of the instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. `*****` indicates the password used for logging in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

You have successfully accessed the instance if the following command output is displayed. Enter commands to perform read and write operations on the database.

Figure 4-5 Redis connected successfully

```
>>> r = redis.StrictRedis(host='192.168.0.143', port=6379, password='*****');
>>> r.set("foo", "bar")
True
>>> print(r.get("foo"))
b'bar'
>>> _
```

- If the instance is a Redis Cluster instance:
 - a. Install the redis-py-cluster client.
 - i. Download the released version.
wget https://github.com/Grokzen/redis-py-cluster/releases/download/2.1.3/redis-py-cluster-2.1.3.tar.gz
 - ii. Decompress the package.
tar -xvf redis-py-cluster-2.1.3.tar.gz
 - iii. Go to the directory where the decompressed redis-py-cluster package is saved, and install redis-py-cluster.

python setup.py install

- b. Access the DCS Redis instance by using redis-py-cluster.

In the following steps, commands are executed in CLI mode.

(Alternatively, write the commands into a Python script and then execute the script.)

- i. Run the **python** command to enter the CLI mode.
- ii. Run the following command to access the chosen DCS Redis instance. If the instance does not have a password, exclude **password='*****'** from the command.

```
>>> from rediscluster import RedisCluster
```

```
>>> startup_nodes = [{"host": "192.168.0.143", "port": "6379"}, {"host": "192.168.0.144",
```

```
"port": "6379"}, {"host": "192.168.0.145", "port": "6379"}, {"host": "192.168.0.146", "port": "6379"}]

>>> rc = RedisCluster(startup_nodes=startup_nodes, decode_responses=True,
password='*****')

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

----End

4.3.3 go-redis

Access a DCS Redis instance through go-redis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed.
For details, see [Viewing Instance Details](#).
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).

Procedure

Step 1 Log in to the ECS.

A Windows ECS is used as an example.

Step 2 Install Visual Studio Community 2017 on the ECS.

Step 3 Start Visual Studio and create a project. The project name can be customized. In this example, the project name is set to **redisdemo**.

Step 4 Import the dependency package of go-redis and enter **go get github.com/go-redis/redis** on the terminal.

Step 5 Write the following code:

```
package main

import (
    "fmt"
    "github.com/go-redis/redis"
)

func main() {
    // Single-node
    rdb := redis.NewClient(&redis.Options{
        Addr: "host:port",
        Password: "*****", // no password set
        DB: 0, // use default DB
    })

    val, err := rdb.Get("key").Result()
    if err != nil {
        if err == redis.Nil {
```

```
        fmt.Println("key does not exists")
        return
    }
    panic(err)
}
fmt.Println(val)

//Cluster
rdbCluster := redis.NewClusterClient(&redis.ClusterOptions{
    Addrs: []string{"host:port"},
    Password: "*****",
})
val1, err1 := rdbCluster.Get("key").Result()
if err1 != nil {
    if err == redis.Nil {
        fmt.Println("key does not exists")
        return
    }
    panic(err)
}
fmt.Println(val1)
}
```

host:port are the IP address/domain name and port number of the DCS Redis instance. For details about how to obtain the IP address/domain name and port, see [Prerequisites](#). Change them as required. ******* indicates the password used to log in to the DCS Redis instance. This password is defined during DCS Redis instance creation.

Step 6 Run the `go build -o test main.go` command to package the code into an executable file, for example, `test`.

⚠ CAUTION

To run the package in the Linux OS, set the following parameters before packaging:

set GOARCH=amd64

set GOOS=linux

Step 7 Run the `./test` command to access the DCS instance.

----End

4.3.4 hiredis in C++

Access a DCS Redis instance through hiredis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

📖 NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To use C++ to connect to a Redis Cluster instance, see the [C++ Redis client description](#).

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).

- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

Procedure

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

The following uses CentOS as an example to describe how to access an instance in C++.

Step 3 Install GCC, Make, and hiredis.

If the system does not provide a compiling environment, run the following **yum** command to install the environment:

```
yum install gcc make
```

Step 4 Run the following command to download and decompress the hiredis package:

```
wget https://github.com/redis/hiredis/archive/master.zip
```

```
unzip master.zip
```

Step 5 Go to the directory where the decompressed hiredis package is saved, and compile and install hiredis.

```
make
```

```
make install
```

Step 6 Access the DCS instance by using hiredis.

The following describes connection and password authentication of hiredis. For more information on how to use hiredis, visit the Redis official website.

1. Edit the sample code for connecting to a DCS instance, and then save the code and exit.

vim connRedis.c

Example:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
    unsigned int j;
    redisContext *conn;
    redisReply *reply;
    if (argc < 3) {
        printf("Usage: example {instance_ip_address} 6379 {password}\n");
        exit(0);
    }
    const char *hostname = argv[1];
    const int port = atoi(argv[2]);
    const char *password = argv[3];
    struct timeval timeout = { 1, 500000 }; // 1.5 seconds
    conn = redisConnectWithTimeout(hostname, port, timeout);
    if (conn == NULL || conn->err) {
```

```
if (conn) {
    printf("Connection error: %s\n", conn->errstr);
    redisFree(conn);
} else {
    printf("Connection error: can't allocate redis context\n");
}
exit(1);
}
/* AUTH */
reply = redisCommand(conn, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);

/* Set */
reply = redisCommand(conn, "SET %s %s", "welcome", "Hello, DCS for Redis!");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);

/* Get */
reply = redisCommand(conn, "GET welcome");
printf("GET welcome: %s\n", reply->str);
freeReplyObject(reply);

/* Disconnects and frees the context */
redisFree(conn);
return 0;
}
```

2. Run the following command to compile the code:

```
gcc connRedis.c -o connRedis -I /usr/local/include/hiredis -lhiredis
```

If an error is reported, locate the directory where the **hiredis.h** file is saved and modify the compilation command.

After the compilation, an executable **connRedis** file is obtained.

3. Run the following command to access the chosen DCS Redis instance:

```
./connRedis {redis_instance_address} 6379 {password}
```

{redis_instance_address} indicates the IP address/domain name of DCS instance and **6379** is an example port number of DCS instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. *{password}* indicates the password used to log in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

You have successfully accessed the instance if the following command output is displayed:

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

NOTICE

If an error is reported, indicating that the hiredis library files cannot be found, run the following commands to copy related files to the system directories and add dynamic links:

```
mkdir /usr/lib/hiredis
cp /usr/local/lib/libhiredis.so.0.13 /usr/lib/hiredis/
mkdir /usr/include/hiredis
cp /usr/local/include/hiredis/hiredis.h /usr/include/hiredis/
echo '/usr/local/lib' >> >> /etc/ld.so.conf
ldconfig
```

Replace the locations of the `so` and `.h` files with actual ones before running the commands.

----End

4.3.5 C#

Access a DCS Redis instance through C# Client StackExchange.Redis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

NOTE

If you use the StackExchange client to connect to a Proxy Cluster instance, the multi-DB function cannot be used.

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

Procedure

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

A Windows ECS is used as an example.

Step 3 Install Visual Studio Community 2017 on the ECS.

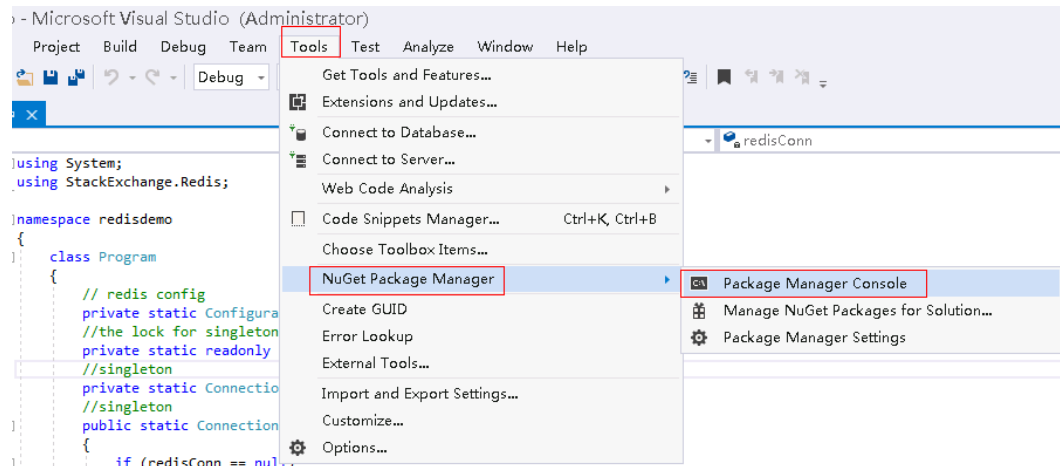
Step 4 Start Visual Studio 2017 and create a project.

Set the project name to **redisdemo**.

Step 5 Install StackExchange.Redis by using the NuGet package manager of Visual Studio.

Access the NuGet package manager console according to [Figure 4-6](#), and enter **Install-Package StackExchange.Redis -Version 2.2.79**. (The version number is optional).

Figure 4-6 Accessing the NuGet package manager console



Step 6 Write the following code, and use the String Set and Get methods to test the connection.

```
using System;
using StackExchange.Redis;

namespace redisdemo
{
    class Program
    {
        // redis config
        private static ConfigurationOptions connDCS = ConfigurationOptions.Parse("{instance_ip_address}:
{port},password=*****,connectTimeout=2000");
        //the lock for singleton
        private static readonly object Locker = new object();
        //singleton
        private static ConnectionMultiplexer redisConn;
        //singleton
        public static ConnectionMultiplexer getRedisConn()
        {
            if (redisConn == null)
            {
                lock (Locker)
                {
                    if (redisConn == null || !redisConn.IsConnected)
                    {
                        redisConn = ConnectionMultiplexer.Connect(connDCS);
                    }
                }
            }
            return redisConn;
        }
        static void Main(string[] args)
        {
            redisConn = getRedisConn();
            var db = redisConn.GetDatabase();
            //set get
            string strKey = "Hello";
            string strValue = "DCS for Redis!";
            Console.WriteLine( strKey + " , " + db.StringGet(strKey));
        }
    }
}
```

```
        Console.ReadLine();  
    }  
}
```

{instance_ip_address} and *{port}* are the IP address/domain name and port number of the DCS Redis instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. ******* indicates the password used for logging in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

Step 7 Run the code. You have successfully accessed the instance if the following command output is displayed:

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

For more information about other commands of StackExchange.Redis, visit [StackExchange.Redis](#).

----End

4.3.6 PHP

4.3.6.1 phpredis

Access a DCS Redis instance through phpredis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To use phpredis to connect to a Redis Cluster instance, see the [phpredis description](#).

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

Procedure

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

The following uses CentOS as an example to describe how to access an instance through phpredis.

Step 3 Install GCC-C++ and Make compilation components.

```
yum install gcc-c++ make
```

Step 4 Install the PHP development package and CLI tool.

Run the following **yum** command to install the PHP development package:

```
yum install php-devel php-common php-cli
```

After the installation is complete, run the following command to query the PHP version and check whether the installation is successful:

```
php --version
```

Step 5 Install the phpredis client.

1. Download the source phpredis package.

```
wget http://pecl.php.net/get/redis-5.3.7.tgz
```

This version is used as an example. To download phpredis clients of other versions, visit the Redis or PHP official website.

2. Decompress the source phpredis package.

```
tar -zxvf redis-5.3.7.tgz  
cd redis-5.3.7
```

3. Command before compilation.

```
phpize
```

4. Configure the **php-config** file.

```
./configure --with-php-config=/usr/bin/php-config
```

The location of the file varies depending on the OS and PHP installation mode. You are advised to locate the directory where the file is saved before the configuration.

```
find / -name php-config
```

5. Compile and install the phpredis client.

```
make && make install
```

6. After the installation, add the **extension** configuration in the **php.ini** file to reference the Redis module.

```
vim /etc/php.ini
```

Add the following configuration:

```
extension = "/usr/lib64/php/modules/redis.so"
```

NOTE

The **redis.so** file may be saved in a different directory from **php.ini**. Run the following command to locate the directory:

```
find / -name php.ini
```

7. Save the configuration and exit. Then, run the following command to check whether the extension takes effect:

```
php -m |grep redis
```

If the command output contains **redis**, the phpredis client environment has been set up.

Step 6 Access the DCS instance by using phpredis.

1. Edit a **redis.php** file.

```
<?php  
$redis_host = "{redis_instance_address}";
```

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

{redis_instance_address} indicates the example IP address/domain name of the DCS instance and *{port}* indicates the port number of the DCS instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. *{password}* indicates the password used to log in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation. If password-free access is enabled, shield the **if** statement for password authentication.

2. Run the **php redis.php** command to access the DCS instance.

----End

4.3.6.2 Predis

Access a DCS Redis instance through Predis on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the PHP compilation environment has been installed on the ECS.

Procedure

- Step 1** View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

- Step 2** Log in to the ECS.

- Step 3** Install the PHP development package and CLI tool. Run the following **yum** command:

```
yum install php-devel php-common php-cli
```

- Step 4** After the installation is complete, check the version number to ensure that the installation is successful.

```
php --version
```

Step 5 Download the Predis package to the `/usr/share/php` directory.

1. Run the following command to download the Predis source file:

```
wget https://github.com/predis/predis/archive/refs/tags/v2.2.2.tar.gz
```

 **NOTE**

This version is used as an example. To download Predis clients of other versions, visit the Redis or PHP official website.

2. Run the following commands to decompress the source Predis package:

```
tar -zxvf predis-2.2.2.tar.gz
```

3. Rename the decompressed Predis directory **predis** and move it to `/usr/share/php/`.

```
mv predis-2.2.2 predis
```

Step 6 Edit a file used to connect to Redis.

- Example of using **redis.php** to connect to a single-node, master/standby, or Proxy Cluster DCS Redis instance:

```
<?php
require 'predis/autoload.php';
Predis\Autoloader::register();
$client = new Predis\Client([
    'scheme' => 'tcp',
    'host'   => '{redis_instance_address}',
    'port'   => {port},
    'password' => '{password}'
]);
$client->set('foo', 'bar');
$value = $client->get('foo');
echo $value;
?>
```

- Example code for using **redis-cluster.php** to connect to Redis Cluster:

```
<?php
require 'predis/autoload.php';
$servers = array(
    'tcp://{redis_instance_address}:{port}'
);
$options = array('cluster' => 'redis');
$client = new Predis\Client($servers, $options);
$client->set('foo', 'bar');
$value = $client->get('foo');
echo $value;
?>
```

{redis_instance_address} indicates the actual IP address/domain name of the DCS instance and *{port}* is the actual port number of DCS instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. *{password}* indicates the password used to log in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation. If password-free access is required, delete the line that contains "password".

Step 7 Run the **php redis.php** command to access the DCS instance.

----End

4.3.7 Node.js

Access a DCS Redis instance through Node.js on an ECS in the same VPC. For more information about how to use other Redis clients, visit [the Redis official website](#).

 NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To use Node.js to connect to a Redis Cluster instance, see [Node.js Redis client description](#).

Prerequisites

- A DCS Redis instance has been created and is in the **Running** state.
- An ECS has been created. For details about how to create an ECS, see [Purchasing an ECS](#).
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

Procedure

- **For client servers running Ubuntu (Debian series):**

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

Step 3 Install Node.js.

```
apt install nodejs-legacy
```

If the preceding command does not work, run the following commands:

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

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

```
cd node-v4.28.5
```

```
./configure
```

```
make
```

```
make install
```

 NOTE

After the installation is complete, run the `node --version` command to query the Node.js version to check whether the installation is successful.

Step 4 Install the node package manager (npm).

```
apt install npm
```

Step 5 Install the Redis client ioredis.

```
npm install ioredis
```

Step 6 Edit the sample script for connecting to a DCS instance.

Add the following content to the `ioRedisdemo.js` script, including information about connection and data reading.

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

host indicates the example IP address/domain name of the DCS instance and *port* indicates the port number of the DCS instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. ******* indicates the password used for logging in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

Step 7 Run the sample script to access the chosen DCS instance.

```
node ioredisdemo.js
```

```
----End
```

- **For client servers running CentOS (Red Hat series):**

Step 1 View the IP address/domain name and port number of the DCS Redis instance to be accessed.

For details, see [Viewing Instance Details](#).

Step 2 Log in to the ECS.

Step 3 Install Node.js.

```
yum install nodejs
```

If the preceding command does not work, run the following commands:

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

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

```
cd node-v0.12.4
```

```
./configure
```

```
make
```

```
make install
```

NOTE

After the installation is complete, run the `node --version` command to query the Node.js version to check whether the installation is successful.

Step 4 Install npm.

```
yum install npm
```

Step 5 Install the Redis client ioredis.

```
npm install ioredis
```

Step 6 Edit the sample script for connecting to a DCS instance.

Add the following content to the **ioredisdemo.js** script, including information about connection and data reading.

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

host indicates the example IP address/domain name of the DCS instance and *port* indicates the port number of the DCS instance. For details about how to obtain the IP address/domain name and port, see [Step 1](#). Change them as required. ******* indicates the password used for logging in to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

Step 7 Run the sample script to access the chosen DCS instance.

```
node ioredisdemo.js
```

```
----End
```

4.4 Connecting to Redis on the Console

Access a DCS Redis instance through Web CLI.

NOTE

- Do not enter sensitive information in Web CLI to avoid disclosure.
- If the value is empty, **nil** is returned after the **GET** command is executed.
- Some commands cannot be run on Web CLI. For details, see [Web CLI Commands](#).

Prerequisites

The instance is in the **Running** state.

Procedure


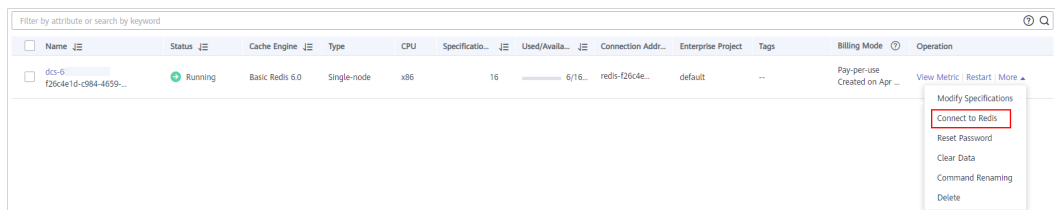
- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**. In the **Operation** column of the instance, choose **More > Connect to Redis**, as shown in the following figure.

Figure 4-7 Accessing Web CLI



- Step 4** Enter the access password of the DCS instance. On Web CLI, select the current Redis database, enter a Redis command in the command box, and press **Enter**.

NOTE

- If no operation is performed for more than 5 minutes, the connection times out. You must enter the access password to connect to the instance again.
- You do not need to enter a password for accessing a password-free DCS Redis instance.

----End


5 Operating DCS Instances

5.1 Viewing Instance Details

On the DCS console, you can view DCS instance details.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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





Step 4 Search for DCS instances using any of the following methods:


- Search by keyword.
Enter a keyword to search.
- Select attributes and enter their keywords to search.
Currently, you can search by name, specification, ID, IP address, AZ, status, instance type, and cache engine.
For example, click the search box, choose **Cache Engine**, and then choose **Redis 4.0**, **Redis 5.0**, or **Redis 6.0**.

For more information on how to search, click the question mark to the right of the search box.

Step 5 Click the name of the DCS instance to display more details about the DCS instance. [Table 5-1](#) describes the parameters.

Table 5-1 Parameters on the Basic Information page of a DCS instance

Section	Parameter	Description
Instance Details	Name	Name of the chosen instance. To modify the instance name, click the  icon.
	Status	State of the chosen instance.
	ID	ID of the chosen instance.
	Cache Engine	Cache engine type of DCS. For example, Redis 4.0.
	Instance Type	Type of the selected instance. Currently, supported types include single-node, master/standby, Proxy Cluster, read/write splitting, and Redis Cluster.
	Cache Size	Specification of the chosen instance.
	Bandwidth	Bandwidth of the DCS instance.
	Used/ Available Memory (MB)	The used memory space and maximum available memory space of the chosen instance. The used memory space includes: <ul style="list-style-type: none"> • Size of data stored on the DCS instance • Size of Redis-server buffers (including client buffer and repl-backlog) and internal data structures
	CPU	CPU architecture of the chosen instance.
	Enterprise Project	Enterprise project to which the new instance belongs. Click  to modify the enterprise project of the instance.
	Maintenance	Time range for any scheduled maintenance activities on cache nodes of this DCS instance. To modify the window, click the  icon.
	Description	Description of the chosen DCS instance. To modify the description, click the  icon.
Connection	Password Protected	Password-protected or password-free access.

Section	Parameter	Description
	Connection Address	<p>Domain name and port number of the instance. You can click  next to Connection Address to change the port.</p> <p>NOTE</p> <ul style="list-style-type: none"> For a master/standby DCS Redis 4.0/5.0/6.0 instance, Connection Address indicates the domain name and port number of the master node, and Read-only Address indicates those of the standby node. When connecting to such an instance, you can use the domain name and port number of the master node or the standby node. For details, see the architecture of a master/standby instance. You can change the port only for a DCS Redis 4.0, 5.0, or 6.0 basic instance, but not for a DCS Redis 3.0, 6.0 professional, or Memcached instance.
	IP Address	IP address and port number of the instance. The domain name connection address is recommended.
Network	AZ	Availability zone in which the cache nodes running the selected DCS instance reside.
	VPC	VPC in which the chosen instance resides.
	Subnet	Subnet in which the chosen instance resides.
	Security Group	To control access to DCS Redis instances, configure a whitelist
Instance Topology	-	<p>Hover over a node to view its metrics, or click the icon of a node to view its historical metrics.</p> <p>Single-node instances do not display the instance topology.</p>
Billing	Billing Mode	Billing mode of the instance.
	Created	Time at which the chosen instance started to be created.
	Run	Time at which the instance was created.

----End

5.2 Modifying Specifications

On the DCS console, you can scale a DCS Redis instance to a larger or smaller capacity, or change the instance type.

 NOTE

- **Modify instance specifications during off-peak hours.** If the modification failed in peak hours (for example, when memory or CPU usage is over 90% or write traffic surges), try again during off-peak hours.
- If your DCS instances are too old to support specification modification, contact technical support to upgrade the instances.
- Modifying instance specifications does not affect the connection address, password, data, security group, and whitelist configurations of the instance. You do not need to restart the instance.

Change of the Instance Type

Table 5-2 Instance type change options supported by different DCS instances

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

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

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

Scaling

- **Scaling options**

Table 5-3 Scaling options supported by different instances

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

- **Impact of scaling**


Table 5-4 Impact of scaling

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

Instance Type	Scaling Type	Impact
Proxy Cluster and Redis Cluster	Scaling up/down	<ul style="list-style-type: none"> ● Scaling out by adding shards: <ul style="list-style-type: none"> – Scaling out does not interrupt connections but will occupy CPU resources, decreasing performance by up to 20%. – If the shard quantity increases, new Redis Server nodes are added, and data is automatically balanced to the new nodes, increasing the access latency. ● Scaling in by reducing shards: <ul style="list-style-type: none"> – If the shard quantity decreases, nodes will be deleted. Before scaling in a Redis Cluster instance, ensure that the deleted nodes are not directly referenced in your application, to prevent service access exceptions. – Nodes will be deleted, and connections will be interrupted. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after scaling. ● Scaling up by shard size without changing the shard quantity: Currently unavailable. ● Scaling down by reducing the shard size without changing the shard quantity has no impact. ● To scale down an instance, ensure that the used memory of each node is less than 70% of the maximum memory per node of the new flavor. ● The flavor changing operation may involve data migration, and the latency may increase. For a Redis Cluster instance, ensure that the client can process the MOVED and ASK commands. Otherwise, the request will fail. ● If the memory becomes full during scaling due to a large amount of data being written, scaling will fail. ● Before scaling, check for big keys through Cache Analysis. Redis has a limit on key migration. If the instance has any single key greater than 512 MB, scaling will fail when big key migration between nodes times out. The bigger the key, the more likely the migration will fail. ● Before scaling a Redis Cluster instance, ensure that automated cluster topology refresh is enabled. If it is disabled, you will need to restart the client after scaling. For details about how to enable automated refresh if you use Lettuce, see an example of using Lettuce to connect to a Redis Cluster instance.

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

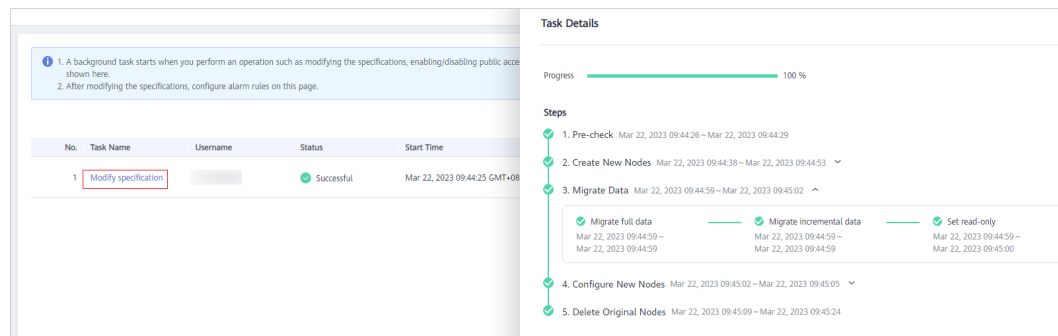
Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Choose **More > Modify Specifications** in the row containing the DCS instance.
- Step 5** On the **Modify Specifications** page, select the desired specification.
- Step 6** Click **Next**, confirm the details, and click **Submit**.

You can go to **Background Tasks** page to view the modification status. For more information, see [Viewing Background Tasks](#).

Specification modification of a single-node, master/standby, or read/write splitting DCS instance takes about 5 to 30 minutes to complete, while that of a cluster DCS instance takes a longer time. After an instance is successfully modified, it changes to the **Running** state.

Go to the **Background Tasks** page to view task details.

Figure 5-1 Viewing background task details**NOTE**

- If the specification modification of a single-node DCS instance fails, the instance is temporarily unavailable for use. The specification remains unchanged. Some management operations (such as parameter configuration and specification modification) are temporarily not supported. After the specification modification is completed in the backend, the instance changes to the new specification and becomes available for use again.
- If the specification modification of a master/standby or cluster DCS instance fails, the instance is still available for use with its original specifications. Some management operations (such as parameter configuration, backup, restoration, and specification modification) are temporarily not supported. Remember not to read or write more data than allowed by the original specifications; otherwise, data loss may occur.
- After the specification modification is successful, the new specification of the instance takes effect.

----End

5.3 Restarting an Instance

On the DCS console, you can restart one or multiple DCS instances at a time.


NOTICE

- After a single-node DCS instance is restarted, data will be deleted from the instance.
- While a DCS instance is restarting, it cannot be read or written.
- An attempt to restart a DCS instance while it is being backed up may result in a failure.
- Restarting a DCS instance will disconnect the original client. You are advised to configure automatic reconnection in your application.

Prerequisites

The DCS instances you want to restart are in the **Running** or **Faulty** state.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** On the **Cache Manager** page, select one or more DCS instances you want to restart.
- Step 5** Click **Restart** above the DCS instance list.
- Step 6** In the displayed dialog box, click **Yes**.

It takes **10 seconds to 30 minutes** to restart DCS instances. After DCS instances are restarted, their status changes to **Running**.

NOTE

- To restart a single instance, you can also click **Restart** in the row containing the desired instance.
- The time required for restarting a DCS instance depends on the cache size of the instance.

----End

5.4 Deleting an Instance

On the DCS console, you can delete one or multiple DCS instances at a time. You can also delete all instance creation tasks that have failed to run.

NOTICE

- After a DCS instance is deleted, the instance data will be deleted without backup. In addition, any backup data of the instance will be deleted. Therefore, download the backup files of the instance for permanent storage before deleting the instance.
 - If the instance is in cluster mode, all cluster nodes will be deleted.
 - Instances billed on a yearly/monthly basis cannot be deleted.
-


Prerequisites

- The DCS instances you want to delete have been created.
- The DCS instances you want to delete are in the **Running**, **Faulty**, or **Stopped** state.

Procedure

Deleting DCS Instances

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 On the **Cache Manager** page, select one or more DCS instances you want to delete.

DCS instances in the **Creating, Starting, Stopping, or Restarting** state cannot be deleted.

Step 5 Choose **More > Delete** above the instance list.

Step 6 Enter **DELETE** and click **Yes** to delete the DCS instance.

It takes 1 to 30 minutes to delete DCS instances.


 **NOTE**

To delete a single instance, choose **More > Delete** in **Operation** column in the row containing the instance.

----End

Deleting Instance Creation Tasks That Have Failed to Run

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

If there are DCS instances that have failed to be created, **Instance Creation Failures** and the number of instances that fail to be created is displayed above the instance list.

Step 4 Click the icon or the number of failed tasks next to **Instance Creation Failures**.

The **Instance Creation Failures** dialog box is displayed.

Step 5 Delete failed instance creation tasks as required.

- To delete all failed tasks, click **Delete All** above the task list.
- To delete a single failed task, click **Delete** in the row containing the task.

----End

5.5 Performing a Master/Standby Switchover

On the DCS console, you can manually switch the master and standby nodes of a master/standby or read/write splitting DCS instance. This operation is used for special purposes, for example, releasing all service connections or terminating ongoing service operations.

This operation is not available for cluster instances. To perform a manual switchover for a Proxy Cluster or Redis Cluster instance, go to the **Node Management** or **Shards and Replicas** page of the instance. For details, see [Managing Nodes](#).


NOTICE

- During a master/standby switchover, services will be interrupted for up to 10 seconds. Before performing this operation, ensure that your application supports connection re-establishment in case of disconnection.
 - During a master/standby node switchover, a large amount of resources will be consumed for data synchronization between the master and standby nodes. You are advised to perform this operation during off-peak hours.
 - Data of the maser and standby nodes is synchronized asynchronously. Therefore, a small amount of data that is being operated on during the switchover may be lost.
 - The instance IP address does not change after a master/standby switchover, so the client does not need to change the connection address.
-

Prerequisites

The DCS instance for which you want to perform a master/standby switchover is in the **Running** state.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** In the **Operation** column of the instance, choose **More > Master/Standby Switchover**.

----End


5.6 Clearing DCS Instance Data

On the DCS console, you can clear data for DCS Redis instances. Clearing instance data cannot be undone and cleared data cannot be recovered. Exercise caution when performing this operation.

Prerequisites

The DCS Redis 4.0/5.0/6.0 instance is in the **Running** state.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Select one or more DCS instances.
- Step 5** Choose **More > Clear** above the instance list.
- Step 6** In the displayed dialog box, click **Yes**.

----End

5.7 Exporting Instance List

On the DCS console, you can export DCS instance information in full to an Excel file.

Procedure


- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** On the **Cache Manager** page, click **Export**. The **Tasks** page is displayed.
- Step 5** When the **Export DCS instance list** task is in the **Successful** state, click **Download** on the right side.

Figure 5-2 Exported DCS instance list

Name	ID	Status	AZ	Cache Engine	Instance Specifics	Used/Avail	Connectivity	Created	Billing	IVPC	VPC ID	Enterprise Project	
dc5-trpt	5e4f4c58	Running	AZ1	Redis 5.0	Single-node	0.125/0.128	0%	198.19.32	May 24, 2	Free	null	null	default
dc5-APITe	693491b0	Running		Redis 3.0	Master/Slave	2/2	1,536	172.16.14	May 06, 2	Yearly	Mcnull	52267da0	default


----End

5.8 Renaming Commands

Redis instances support command renaming. Currently, you can only rename the **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, **HGETALL**, **SCAN**, **HSCAN**, **SSCAN**, and **ZSCAN** commands. For Proxy Cluster instances, you can also rename the **DBSIZE** and **DBSTATS** commands.

Redis 3.0 is no longer available and does not support command renaming.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** In the **Operation** column of an instance, choose **More > Command Renaming**.
- Step 5** Select a command, enter a new name, and click **OK**.

In the **Command Renaming** dialog box, click **Add Command** to rename multiple commands at the same time. After renaming commands, you can view the renaming operation record on the **Background Tasks** page.

Figure 5-3 Renaming commands

No.	Task Name	Username	Status	Start Time	End Time	Detailed information	Operation
1	Command Renaming		● Successful	Jul 17, 2023 16:38:54 GMT+08:00	Jul 17, 2023 16:39:10 GMT+08:00	Before: command	Delete

NOTE

- An instance will restart when its commands are renamed.
- Remember the new command names because they will not be displayed on the console for security purposes.
- A command can be renamed multiple times. Each new name overwrites the previous name.

----End

6 Managing DCS Instances

6.1 Configuration Notice

In most cases, different DCS instance management operations cannot proceed concurrently. If you initiate a new management operation while the current operation is in progress, the DCS console prompts you to initiate the new operation again after the current operation is complete. DCS instance management operations include:

- Creating a DCS instance
- Configuring parameters
- Restarting a DCS instance
- Changing the instance password
- Resetting the instance password
- Scaling, backing up, or restoring an instance

You can restart a DCS instance while it is being backed up, but the backup task will be forcibly interrupted and is likely to result in a backup failure.

NOTICE

In the event that a cache node of a DCS instance is faulty:

- The instance remains in the **Running** state and you can continue to read from and write to the instance. This is achieved thanks to the high availability of DCS.
 - Cache nodes can recover from internal faults automatically. Manual fault recovery is also supported.
 - Certain operations (such as backup, restoration, and parameter configuration) in the management zone are not supported during fault recovery. You can contact customer service or perform these operations after cache nodes recover from faults.
-

6.2 Modifying Configuration Parameters

6.2.1 Modifying Configuration Parameters of an Instance


On the DCS console, you can configure parameters for an instance to achieve optimal DCS performance.

For example, if you do not need data persistence, set **appendonly** to **no**.

 **NOTE**

After the instance configuration parameters are modified, the modification takes effect immediately without the need to manually restart the instance. For a cluster instance, the modification takes effect on all shards.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** On the **Cache Manager** page, click the name of the DCS instance you want to configure.
- Step 5** On the instance details page, choose **Instance Configuration > Parameters**.
- Step 6** Click **Modify**.
- Step 7** Modify parameters based on your requirements.

The parameters are described in [Table 6-1](#). In most cases, you can retain default values.

Table 6-1 DCS Redis instance configuration parameters

Parameter	Description	Value Range	Default Value
timeout	The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of 0 means that this function is disabled. Proxy Cluster instances do not have this parameter.	0-7200 seconds	0
appendfsync	Controls how often fsync() transfers cached data to the disk. Note that some OSs will perform a complete data transfer but some others only make a "best-effort" attempt. Single-node instances do not have this parameter. There are three settings: no: fsync() is never called. The OS will flush data when it is ready. This mode offers the highest performance. always: fsync() is called after every write to the AOF. This mode is very slow, but also very safe. everysec: fsync() is called once per second. This mode provides a compromise between safety and performance.	<ul style="list-style-type: none"> • no • always • everysec 	no

Parameter	Description	Value Range	Default Value
appendonly	<p>Indicates whether to log each modification of the instance. By default, data is written to disks asynchronously in Redis. If this function is disabled, recently-generated data might be lost in the event of a power failure. Single-node instances do not have this parameter.</p> <p>Options:</p> <p>yes: Logs are enabled, that is, persistence is enabled.</p> <p>no: Logs are disabled, that is, persistence is disabled.</p>	<ul style="list-style-type: none"> • yes • no 	yes
client-output-buffer-limit-slave-soft-seconds	Number of seconds that the output buffer remains above client-output-buffer-slave-soft-limit before the client is disconnected.	0-60	60
client-output-buffer-slave-hard-limit	Hard limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected.	0-17,179,869,184	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the soft limit and continuously remains above the limit for the time specified by the client-output-buffer-limit-slave-soft-seconds parameter, the client is disconnected.	0-17,179,869,184	1,717,986,918

Parameter	Description	Value Range	Default Value
maxmemory-policy	<p>The policy applied when the maxmemory limit is reached. Options:</p> <ul style="list-style-type: none"> • volatile-lru: Evict keys by trying to remove the less recently used (LRU) keys first, but only among keys that have an expire set. • allkeys-lru: Evict keys by trying to remove the LRU keys first. • volatile-random: Evict keys randomly, but only among keys that have an expire set. • allkeys-random: Evict keys randomly. • volatile-ttl: Evict keys with an expire set, and try to evict keys with a shorter time to live (TTL) first. • noeviction: Do not delete any keys and only return errors when the memory limit was reached. • volatile-lfu: Evict keys by trying to remove the less frequently used (LFU) keys first, but only among keys that have an expire set. • allkeys-lfu: Evict keys by trying to remove the LFU keys first. <p>For details about eviction policies, see the Redis official website.</p>	<ul style="list-style-type: none"> • volatile-lru • allkeys-lru • volatile-random • allkeys-random • volatile-ttl • noeviction • volatile-lfu • allkeys-lfu 	volatile-lru
lua-time-limit	Maximum time allowed for executing a Lua script (in milliseconds).	100-5000	5000

Parameter	Description	Value Range	Default Value
master-read-only	Sets the instance to be read-only. All write operations will fail. Proxy Cluster instances do not have this parameter.	<ul style="list-style-type: none"> • yes • no 	no
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. This parameter specifies the maximum number of connections on a single node (single shard). <ul style="list-style-type: none"> • Cluster: Maximum connections limit per node = Maximum connections limit of the instance/Shard quantity • Single-node, master/standby, and read/write splitting: Maximum connections limit on a single node = Maximum connections limit of the instance 	1000-50,000	10,000
proto-max-bulk-len	Maximum size of a single element request (in bytes).	1,048,576-536,870,912	536,870,912
repl-backlog-size	The replication backlog size (bytes). The backlog is a buffer that accumulates replica data when replicas are disconnected from the master. When a replica reconnects, a partial synchronization is performed to synchronize the data that was missed while replicas were disconnected.	16,384-1,073,741,824	1,048,576

Parameter	Description	Value Range	Default Value
repl-backlog-ttl	The amount of time, in seconds, before the backlog buffer is released, starting from the last a replica was disconnected. The value 0 indicates that the backlog is never released.	0-604,800	3600
repl-timeout	Replication timeout (in seconds).	30-3600	60
hash-max-ziplist-entries	The maximum number of hashes that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	512
hash-max-ziplist-value	The largest value allowed for a hash encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64
set-max-intset-entries	When a set is composed entirely of strings and number of integer elements is less than this parameter value, the set is encoded using intset, a data structure optimized for memory use.	1-10,000	512
zset-max-ziplist-entries	The maximum number of sorted sets that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	128
zset-max-ziplist-value	The largest value allowed for a sorted set encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64

Parameter	Description	Value Range	Default Value
latency-monitor-threshold	<p>The minimum amount of latency that will be logged as latency spikes</p> <ul style="list-style-type: none"> • Set to 0: Latency monitoring is disabled. • Set to more than 0: All with at least this many ms of latency will be logged. <p>By running the LATENCY command, you can perform operations related to latency monitoring, such as obtaining statistical data, and configuring and enabling latency monitoring.</p> <p>Proxy Cluster instances do not have this parameter.</p>	0-86,400,000 ms	0

Parameter	Description	Value Range	Default Value
notify-keyspace-events	Controls which keyspace events notifications are enabled for. If this parameter is configured, the Redis Pub/Sub feature will allow clients to receive an event notification when a Redis data set is modified. Proxy Cluster instances do not have this parameter.	A combination of different values can be used to enable notifications for multiple event types. Possible values include: K: Keyspace events, published with the __keyspace@*__ prefix E: Keyevent events, published with __keyevent@*__ prefix g: Generic commands (non-type specific) such as DEL, EXPIRE, and RENAME \$: String commands l: List commands s: Set commands h: Hash commands z: Sorted set commands x: Expired events (events generated every time a key expires) e: Evicted events (events generated when a key is evicted from maxmemory) For more information, see the following note.	Ex
slowlog-log-slower-than	The maximum amount of time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis slow query log will record the command.	0-1,000,000	10,000

Parameter	Description	Value Range	Default Value
slowlog-max-len	The maximum allowed number of slow queries that can be logged. Slow query log consumes memory, but you can reclaim this memory by running the SLOWLOG RESET command.	0-1000	128
auto-kill-timeout-lua-process	<p>yes: enable no: disable</p> <p>When this parameter is enabled, lua scripts are killed when their execution times out. However, scripts with write operations are not killed, but their nodes automatically restart (if persistence has been enabled for the instance) without saving the write operations.</p> <p>Single-node instances do not have this parameter.</p>	<ul style="list-style-type: none"> • yes • no 	no

 **NOTE**

1. For more information about the parameters described in [Table 6-1](#), visit <https://redis.io/topics/memory-optimization>.
2. The **latency-monitor-threshold** parameter is usually used for fault location. After locating faults based on the latency information collected, change the value of **latency-monitor-threshold** to **0** to avoid unnecessary latency.
3. More about the **notify-keyspace-events** parameter:
 - The parameter setting must contain at least a **K** or **E**.
 - **A** is an alias for "g\$lshzxe" and cannot be used together with any of the characters in "g\$lshzxe".
 - For example, the value **KI** means that Redis will notify Pub/Sub clients about keyspace events and list commands. The value **AKE** means Redis will notify Pub/Sub clients about all events.
4. Configurable parameters and their values vary depending on the instance type.

Step 8 After you have finished setting the parameters, click **Save**.

Step 9 Click **Yes** to confirm the modification.

----End





6.3 Modifying Maintenance Window

On the DCS console, after creating a DCS instance, you can modify the maintenance window of the DCS instance on the instance's **Basic Information** page. During the maintenance window, O&M personnel can maintain the instance.

Prerequisites

A DCS instance has been created.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of the desired DCS instance.
- Step 5** Click the **Basic Information** tab. In the **Instance Details** area, click the  icon next to the **Maintenance** parameter.
- Step 6** Select a new maintenance window from the drop-down list. Click  to save the modification or  to discard the modification.

The modification will take effect immediately on the **Basic Information** tab page.

NOTE


The duration of each maintenance window is one hour, for example, from 02:00 to 03:00.

----End

6.4 Viewing Background Tasks

After you initiate certain instance operations such as scaling up the instance and changing or resetting a password, a background task will start for each operation. On the DCS console, you can view the background task status and clear task information by deleting task records.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.

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


Filter DCS instances to find the desired DCS instance. Currently, you can search instances by name, specification, ID, IP address, AZ, status, instance type, cache engine, and many other attributes.

Step 4 Click the name of the DCS instance to display more details about the DCS instance.

Step 5 Click the **Background Tasks** tab.

A list of background tasks is displayed.

Step 6 Click  , specify **Start Date** and **End Date**, and click **OK** to view tasks started in the corresponding time segment.

- Click  to refresh the task status.
- To clear the record of a background task, choose **Operation > Delete**.

 **NOTE**

You can only delete the records of tasks in the **Successful** or **Failed** state.

----End


6.5 Managing IP Address Whitelist

The following describes how to manage whitelists of a Redis instance to allow access only from whitelisted IP addresses.

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

Creating a Whitelist Group

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 Click the name of a DCS instance.

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

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

Table 6-2 Whitelist parameters

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

Step 7 Click **OK**.

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

NOTE

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

----End

6.6 Managing Tags

Tags facilitate DCS instance identification and management.


You can add tags to an instance when creating it or add, modify, or delete tags on the details page of a created instance. Each instance can have a maximum of 20 tags.

A tag consists of a tag key and a tag value. [Table 6-3](#) lists the tag key and value requirements.

Table 6-3 Tag key and value requirements

Parameter	Requirements
Tag key	<ul style="list-style-type: none"> • Cannot be left blank. • Must be unique for the same instance. • Consists of a maximum of 128 characters. • Can contain letters of any language, digits, spaces, and special characters <code>_ . : = + - @</code> • Cannot start or end with a space. • Cannot start with <code>_sys_</code>.
Tag value	<ul style="list-style-type: none"> • Consists of a maximum of 255 characters. • Can contain letters of any language, digits, spaces, and special characters <code>_ . : / = + - @</code> • Cannot start or end with a space.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of the desired DCS instance to go to the details page.
- Step 5** Choose **Instance Configuration > Tags**.
- Step 6** Perform the following operations as required:
- Add a tag
 - Click **Add/Edit Tag**.
If you have created predefined tags, select a predefined pair of tag key and value. To view or create predefined tags, click **View predefined tags**. Then you will be directed to the TMS console.
You can also create new tags by specifying **Tag key** and **Tag value**.
 - Click **OK**.
 - Modify a tag
Click **Add/Edit Tag**. In the displayed **Add/Edit Tag** dialog box, delete the desired key, add the key again, enter a new tag value, and click **Add**.
 - Delete a tag

In the row that contains the desired tag, click **Delete**. In the displayed dialog box, click **Yes**.

----End

6.7 Managing Nodes

This section describes how to query the shards and replicas of a master/standby, cluster, or read/write splitting DCS Redis instance, and how to manually promote a replica to master.


- By default, a master/standby or read/write splitting instance has only one shard with one master and one replica. You can view the sharding information on the **Node Management** page. To manually switch the master and replica roles, see [Performing a Master/Standby Switchover](#).
- If a master/standby instance has multiple replicas, you can remove the IP address from a replica and set the failover priority on the **Node Management** page.
- A Proxy Cluster or Redis Cluster instance has multiple shards. Each shard has one master and one replica. On the **Node Management** page, you can view the sharding information and manually switch the master and replica roles.

NOTE

- This feature is called "Node Management" in some regions and "Shard and Replica" in the other regions. Refer to the console for the actual name.
- This feature is supported by DCS Redis 4.0 instances and later.
- For single-node DCS instances, this feature is supported only in regions where **Node Management** is used.
- For details about the number of shards for different instance specifications, see [Redis Cluster](#) and [Proxy Cluster Redis](#).
- You can add shards to a cluster instance by referring to [Modifying Specifications](#).

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 Click an instance.

Step 5 Click **Node Management** or **Shards and Replicas**.

The page displays all shards in the instance and the list of replicas of each shard.


Step 6 Click  to show all replicas of a shard, as shown in the following figure.

Figure 6-1 Node management (cluster instance)

Shard Name	Shard ID	Replicas				
group-0	9c20044-699d-4250-a139-16a1bb0b0ade	2				
192.168.28.12	#0000230be9180018bc85d5c34075	86abc931-10ec-4c0c-8d95-2f862a8baf6	Running	Master	AZ1	
192.168.27.236	#0000230be9180018bc85d5c44076	db48f5a-250c-44be-8064-115522ee570b	Running	Replica	AZ2	Promote to Master
group-1	c3082976-700c-4b56-be3f-23e17818110	2				
group-2	9e87a08c-7700-4212-91e9-1577d966ca3	2				

Figure 6-2 Node management (master/standby instance)

Shard Name	Shard ID	Replicas				
group-0	f5185864-4c2ba-4ca1-a899-0a67468793c8	2				
10.0.0.179	#0000230be9180018bc882a86c400a	60ccedf9-9054-40be-9ca8-678601a96fc	Running	Master	AZ1	
10.0.0.186	#0000230be9180018bc882a878400b	cta2af74-343c-469c-8c29-d6ed1e9967ae	Running	Replica	AZ2	100 Remove IP Address

Figure 6-3 Node management (single-node instance)

Shard Name	Shard ID	Replicas				
group-0	7b2ab30-0b60-45cf-aac1-d03e18400564	1				
10.0.0.38	#0000230b10079010b0b094b4c2d		Running	Master	AZ1	

- Cluster

To promote a replica to the master role, expand a shard and click **Promote to Master** in the row that contains a node whose **Role** is **Replica**.

NOTE

You can view the proxy information of a Proxy Cluster instance on the **Proxies** tab page only in regions where **Node Management** is used. Other types of instances do not have the **Proxies** tab page.

- Master/Standby or read/write splitting

- If a master/standby instance has multiple replicas, click **Remove IP Address** in the row containing a read-only replica. After a replica IP address is removed, the read-only domain name will no longer be resolved to the replica IP address.

If a master/standby instance has only one replica, its IP address cannot be removed.

- If a master/standby or read/write splitting instance has multiple replicas, click in the **Failover Priority** column to change the priority of the replica to be promoted to master.

If the master fails, the replica with the smallest priority number is automatically promoted to master. For multiple replicas that have the same priority, a selection process will be performed. **0** indicates that the replica will never be automatically promoted, **1** indicates the highest priority, and **100** indicates the lowest priority.

- Single-node

A single-node instance has only one replica. You can view its node information on the **Node Management** page.

----End

6.8 Cache Analysis

6.8.1 Analyzing Big Keys and Hot Keys

By performing big key analysis and hot key analysis, you will have a picture of keys that occupy a large space and keys that are the most frequently accessed.

Notes on big key analysis:

- All DCS Redis instances support big key analysis.
- During big key analysis, all keys will be traversed. The larger the number of keys, the longer the analysis takes.
- Perform big key analysis during off-peak hours and avoid automatic backup periods.
- For a master/standby or cluster instance, the big key analysis is performed on the standby node, so the impact on the instance is minor. For a single-node instance, the big key analysis is performed on the only node of the instance and will reduce the instance access performance by up to 10%. Therefore, perform big key analysis on single-node instances during off-peak hours.
- A maximum of 100 big key analysis records (20 for Strings and 80 for Lists/Sets/Zsets/Hashes) are retained for each instance. When this limit is reached, the oldest records will be deleted to make room for new records. You can also manually delete records you no longer need.

Notes on hot key analysis:


- The **maxmemory-policy** parameter of the instance must be set to **allkeys-lfu** or **volatile-lfu**.
- During hot key analysis, all keys will be traversed. The larger the number of keys, the longer the analysis takes.
- Perform hot key analysis shortly after peak hours to ensure the accuracy of the analysis results.
- The hot key analysis is performed on the master node of each instance and will reduce the instance access performance by up to 10%.
- A maximum of 100 analysis records are retained for each instance. When this limit is reached, the oldest records will be deleted to make room for new records. You can also manually delete records you no longer need.

NOTE

Perform big key and hot key analysis during off-peak hours to avoid 100% CPU usage.

Procedure for Big Key Analysis

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of a DCS Redis instance.
- Step 5** Choose **Analysis and Diagnosis > Cache Analysis**.
- Step 6** On the **Big Key Analysis** tab page, you can manually start a big key analysis or schedule a daily automatic analysis.
- Step 7** After an analysis task completes, click **View** to view the analysis results of different data types.

You can also click **Download** or **Delete** in the **Operation** column to download or delete the analysis result.

 **NOTE**


The console displays a maximum of 20 big key analysis records for Strings and 80 for Lists, Sets, Zsets, and Hashes.

Table 6-4 Results of big key analysis

Parameter	Description
Key	Name of a big key.
Type	Type of a big key, which can be string, list, set, zset, or hash.
Size	Size or number of elements of a big key.
Database	Database where a big key is located.

----End

Procedure for Hot Key Analysis

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of a DCS Redis instance.
- Step 5** Choose **Analysis and Diagnosis > Cache Analysis**.
- Step 6** On the **Hot Key Analysis** tab page, you can manually start a hot key analysis or schedule a daily automatic analysis.

 NOTE

To perform hot key analysis, set this parameter to **allkeys-lfu** or **volatile-lfu** on the **Instance Configuration > Parameters** page. For details about **allkeys-lfu** and **volatile-lfu**, see [What Is the Default Data Eviction Policy?](#)

Step 7 After an analysis task completes, click **View** to view the analysis results.

You can also click **Download** or **Delete** in the **Operation** column to download or delete the analysis result.

 NOTE

The console displays a maximum of 100 hot key analysis records for each instance.

Table 6-5 Results of hot key analysis

Parameter	Description
Key	Name of a hot key.
Type	Type of a hot key, which can be string, hash, list, set, or sorted set.
Size	Size of the hot key value.
FREQ	Reflects the access frequency of a key within a specific period of time (usually 1 minute). FREQ is the logarithmic access frequency counter. The maximum value of FREQ is 255, which indicates 1 million access requests. After FREQ reaches 255 , it will no longer increment even if access requests continue to increase. FREQ will decrement by 1 for every minute during which the key is not accessed.
Shard	Shard where the hot key is located. NOTE This parameter is available only for cluster instances.
Database	Database where a hot key is located.

----End

FAQs About Big Keys and Hot Keys

- [Why Is the Capacity or Performance of a Shard of a Redis Cluster Instance Overloaded When That of the Instance Is Still Below the Bottleneck?](#)
- [What Are Big Keys and Hot Keys?](#)
- [What Is the Impact of a Hot Key?](#)
- [How Do I Avoid Big Keys and Hot Keys?](#)
- [How Do I Detect Big Keys and Hot Keys in Advance?](#)

6.8.2 Scanning Expired Keys

Background

There are two ways to delete a key in Redis.

- Use the **DEL** command to directly delete a key.
- Use commands such as **EXPIRE** to set a timeout on a key. After the timeout elapses, the key becomes inaccessible but is not deleted immediately because Redis is mostly single-threaded. Redis uses the following strategies to release the memory used by expired keys:
 - Lazy free deletion: The deletion strategy is controlled in the main I/O event loop. Before a read/write command is executed, a function is called to check whether the key to be accessed has expired. If it has expired, it will be deleted and a response will be returned indicating that the key does not exist. If the key has not expired, the command execution resumes.
 - Scheduled deletion: A time event function is executed at certain intervals. Each time the function is executed, a random collection of keys are checked, and expired keys are deleted. (By default, 10 checks are executed every second. Each check randomly scans 20 keys which are set to expire.)

NOTE

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


Expired DCS Key Scan

DCS integrates these strategies and allows you to periodically release the memory used by expired keys. You can configure scheduled scans on the master nodes of your instances. The entire key space is traversed during the scans, triggering Redis to check whether the keys have expired and to remove expired keys if any.

NOTE

- Perform expired key scans during off-peak hours to avoid 100% CPU usage.
- Released expired keys cannot be queried.

Procedure

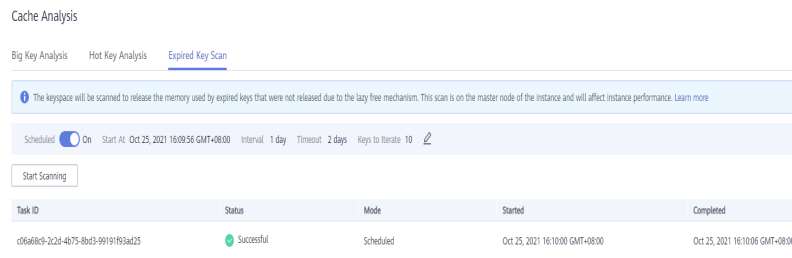
- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of a DCS Redis instance.
- Step 5** Choose **Analysis and Diagnosis > Cache Analysis**.

Step 6 On the **Expired Key Scan** tab page, scan for expired keys and release them.

- Click **Start Scanning** to scan for expired keys immediately.
- Enable **Scheduled** to schedule automatic scans at a specified time. For details about how to configure automatic scans, see [Scheduling Automatic Scans](#).

Step 7 After the expired key scan task is submitted, view it in the task list.

Figure 6-4 Expired key scan tasks



----End

NOTE

The scan fails in the following scenarios:

- An exception occurred.
- There are too many keys, resulting in a timeout. Some keys have already been deleted before the timeout.

Scheduling Automatic Scans


To schedule automatic scans, click  next to **Scheduled**. Set the parameters as required, and click **OK**.

Table 6-6 describes the parameters for scheduling automatic scans.

Table 6-6 Parameters for scheduling automatic scans

Parameter	Description	Value Range	Default Value	Remarks
Start At	The first scan can only start after the current time.	Format: MM/DD/YYYY hh:mm:ss	-	-

Parameter	Description	Value Range	Default Value	Remarks
Interval	Interval between scans.	0 to 43,200 (unit: minute)	1440	<ul style="list-style-type: none"> • If the previous scan is not complete when the start time arrives, the upcoming scan will be skipped. • If the previous scan is complete within five minutes after the start time, the upcoming scan will not be skipped. <p>NOTE Continuous scans may cause high CPU usage. Set this parameter based on the total number of keys in the instance and the increase of keys. For details, see the following performance description and configuration suggestions.</p>

Parameter	Description	Value Range	Default Value	Remarks
Timeout	This parameter is used to prevent scanning timeout due to unknown reasons. If scanning times out due to unknown reasons, subsequent scheduled tasks cannot be executed. After the specified timeout elapses, a failure message is returned and the next scan will be performed.	1 to 86,400 (unit: minute)	2880	<ul style="list-style-type: none"> Set the timeout to at least twice the interval. You can set a value based on the time taken in previous scans and the maximum timeout that can be tolerated in the application scenario.

Parameter	Description	Value Range	Default Value	Remarks
Keys to Iterate	The SCAN command is used to iterate the keys in the current database. The COUNT option is used to let the user tell the iteration command how many elements should be returned from the dataset in each iteration. For details, see the description of the SCAN command . Iterative scanning can reduce the risks of slowing down Redis when a large number of keys are scanned at a time.	10 to 1000	10	For example, if there are 10 million keys in Redis and the number of keys to iterate is set to 1000, a full scan will be complete after 10,000 iterations.

Performance

- The **SCAN** command is executed at the data plane every 5 ms, that is, 200 times per second. If **Keys to Iterate** is set to **10, 50, 100, or 1000**, 2000, 10,000, 20,000, or 200,000 keys are scanned per second.
- The larger the number of keys scanned per second, the higher the CPU usage.

Reference test

A master/standby instance is scanned. There are 10 million keys that will not expire and 5 million keys that will expire. The expiration time is 1 to 10 seconds. A full scan is executed.

 NOTE

The following test results are for reference only. They may vary depending on the site environment and network fluctuation.

- Natural deletion: 10,000 expired keys are deleted per second. It takes 8 minutes to delete 5 million expired keys. The CPU usage is about 5%.
- **Keys to Iterate** set to **10**: The scanning takes 125 minutes (15 million/2000/60 seconds) and the CPU usage is about 8%.
- **Keys to Iterate** set to **50**: The scanning takes 25 minutes (15 million/10,000/60 seconds) and the CPU usage is about 10%.
- **Keys to Iterate** set to **100**: The scanning takes 12.5 minutes (15 million/20,000/60 seconds) and the CPU usage is about 20%.
- **Keys to Iterate** set to **1000**: The scanning takes 1.25 minutes (15 million/200,000/60 seconds) and the CPU usage is about 25%.

Configuration suggestions

- You can configure the number of keys to be scanned and the scanning interval based on the total number of keys and the increase in the number of keys in the instance.
- In the reference test with 15 million keys and **Keys to Iterate** set to **10**, the scanning takes about 125 minutes. In this case, set the scan interval to more than 4 hours.
- If you want to accelerate the scanning, set **Keys to Iterate** to **100**. It takes about 12.5 minutes to complete the scanning. Therefore, set the scan interval to more than 30 minutes.
- The larger the number of keys to iterate, the faster the scanning, and the higher the CPU usage. There is a trade-off between time and CPU usage.
- If the number of expired keys does not increase rapidly, you can scan expired keys once a day.

 NOTE

Start scanning during off-peak hours. Set the interval to one day and the timeout to two days.

6.9 Viewing Redis Slow Queries

Redis logs queries that exceed a specified execution time. You can view the slow logs on the DCS console to identify performance issues.


For details about the commands, visit the [Redis official website](#).

Configure slow queries with the following parameters:

- **slowlog-log-slower-than**: The maximum time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis will log the command. The default value is **10,000**. That is, if command execution exceeds 10 ms, the command will be logged.
- **slowlog-max-len**: The maximum allowed number of slow queries that can be logged. The default value is **128**. That is, if the number of slow queries exceeds 128, the earliest record will be deleted to make room for new ones.

For details about the configuration parameters, see [Modifying Configuration Parameters of an Instance](#).



Viewing Slow Queries on the Console

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of a DCS instance.
- Step 5** Choose **Analysis and Diagnosis > Slow Queries**.
- Step 6** Select a start date and an end date and click the refresh icon to view slow queries within the specified period.

NOTE

- For details about the commands, visit the [Redis official website](#).
- Currently, you can view slow queries in the last seven days.

Figure 6-5 Slow queries of an instance


Executed 	Duration (ms) 	Shard Name	Role	Slow Query
Jul 20, 2023 16:36:46 GMT+08:00	54.51	group-2	Master	keys *
Jul 20, 2023 16:36:46 GMT+08:00	52.374	group-0	Master	keys *
Jul 20, 2023 16:36:46 GMT+08:00	55.282	group-1	Master	keys *
Jul 20, 2023 15:28:51 GMT+08:00	57.545	group-0	Master	keys *
Jul 20, 2023 15:28:51 GMT+08:00	54.134	group-1	Master	keys *

----End

6.10 Viewing Redis Run Logs

You can create run log files on the DCS console to collect run logs of DCS Redis instances within a specified period. After the logs are collected, you can download the log files to view the logs.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.

Step 4 Click a DCS instance.

Step 5 Click the **Run Logs** tab.

Step 6 Click **Create Log File** and specify the collection conditions.

If the instance is the master/standby, read/write splitting, or cluster type, you can specify the shard and replica whose run logs you want to collect. If the instance is the single-node type, logs of the only node of the instance will be collected.

Select the collection period and click **OK**.

Step 7 After the log file is successfully collected, click **Download** to download it.

 **NOTE**

The Redis kernel generates few logs, so your selected period may contain no logs.

----End

6.11 Managing Users

You can create read-only and read/write users to control access permissions.


Prerequisites

ACL is a whitelist feature and is disabled by default. Contact the administrator to enable it.

This function is supported by DCS Redis 4.0/5.0 instances.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 Click an instance.

Step 5 Choose **User Management** in the navigation pane.

The user whose username is **default** is the instance's default user. The default user has read and write permissions and their password is the instance's password.

Step 6 Click **Create User**.

 **NOTE**

- A maximum of 18 users can be created for an instance.
- If **Password Protected** is enabled for a DCS Redis instance, only the default user can be used.
- To use a normal user, click **Reset Password** in the row that contains the default user to disable **Password Protected** for the default user.

Step 7 Specify the **Username** and **Description**. Select **Read-only** or **Read/Write**. Specify the **Password** and confirm it.

Step 8 Click **OK**.

NOTICE

A normal ACL user connects to an instance with password *{username:password}*.

- When **using redis-cli** to connect to an instance, the default user runs the following command:

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

- A normal ACL user runs the following command:

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

----End

More Operations

The following operations can be performed on normal users.

Table 6-7 Operation


Operation	Description
Changing a password	Locate the row that contains the desired normal user and click Change Password in the Operation column.
Reset a password	If password is forgot, locate the row that contains the normal user and click Reset Password in the Operation column.
Modify permissions	Locate the row that contains the normal user. Choose More > Modify Permission in the Operation column. The Read-only or Read/Write permissions can be granted.
Edit description	Locate the row that contains the normal user. Choose More > Edit Description in the Operation column.
Delete a user	Locate the row that contains the normal user. Choose More > Delete in the Operation column.
Batch deleting users	Select the normal users to be deleted and click Delete above the list. The default user cannot be deleted.

6.12 Diagnosing an Instance

Scenario

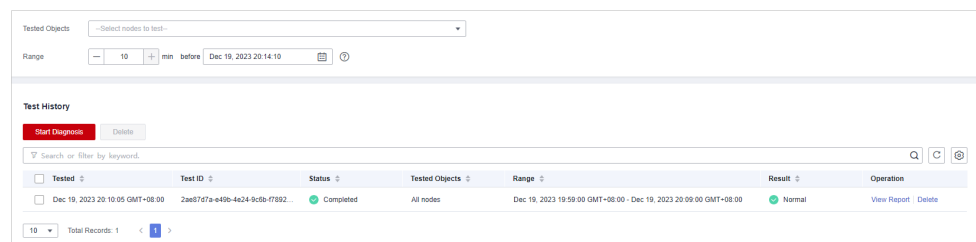
If a fault or performance issue occurs, you can ask DCS to diagnose your instance to learn about the cause and impact of the issue and how to handle it.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the name of a DCS Redis instance.
- Step 5** Choose **Analysis and Diagnosis > Instance Diagnosis**.
- Step 6** Specify the tested object and time range, and click **Start Diagnosis**.
 - **Tested Object:** You can select a single node or all nodes.
 - **Range:** You can specify up to 10 minutes before a point in time in the last 7 days.

The data within 10 minutes before the specified time will be diagnosed as shown below.

Figure 6-6 Specifying the tested object and time range



NOTE

Instance diagnosis may fail during specification modification.

- Step 7** After the diagnosis is complete, you can view the result in the **Test History** list. If the result is abnormal, click **View Report** for details.

In the report, you can view the cause and impact of abnormal items and suggestions for handling them.

----End

7 Backing Up and Restoring Instances

7.1 Overview

On the DCS console, you can back up and restore DCS instances.

Importance of DCS Instance Backup

There is a small chance that dirty data could exist in a DCS instance owing to service system exceptions or problems in loading data from persistence files. In addition, some systems demand not only high reliability but also data security, data restoration, and even permanent data storage.

Currently, data in DCS instances can be backed up to OBS. If a DCS instance becomes faulty, data in the instance can be restored from backup so that service continuity is not affected.

Backup Modes

DCS instances support the following backup modes:

- **Automated backup**

You can create a scheduled backup policy on the DCS console. Then, data in the chosen DCS instances will be automatically backed up at the scheduled time.

You can choose the days of the week on which automated backup will run. Backup data will be retained for a maximum of seven days. Backup data older than seven days will be automatically deleted.

The primary purpose of automated backups is to create complete data replicas of DCS instances so that the instance can be quickly restored if necessary.
- **Manual backup**

Backup requests can be issued manually. Data in the chosen DCS instances will be backed up to OBS.

Before performing high-risk operations, such as system maintenance or upgrade, back up DCS instance data.

Additional Information About Data Backup

- Instance type
 - **Redis: Only master/standby, read/write splitting, Proxy Cluster, and Redis Cluster instances can be backed up and restored, while single-node instances cannot.** You can export data of a single-node instance to an RDB file using redis-cli. For details, see [How Do I Export DCS Redis Instance Data?](#)
- Backup mechanisms

Basic edition DCS for Redis 4.0 and later persist data to RDB or AOF files in manual backup mode, and to RDB files in automatic backup mode.

Backup tasks are run on standby cache nodes. DCS instance data is backed up by compressing and storing the data persistence files from the standby cache node to OBS.

DCS checks instance backup policies once an hour. If a backup policy is matched, DCS runs a backup task for the corresponding DCS instance.
- Impact on DCS instances during backup

Backup tasks are run on standby cache nodes, without incurring any downtime.

In the event of full-data synchronization or heavy instance load, it takes a few minutes to complete data synchronization. If instance backup starts before data synchronization is complete, the backup data will be slightly behind the data in the master cache node.

During instance backup, the standby cache node stops persisting the latest changes to disk files. If new data is written to the master cache node during backup, the backup file will not contain the new data.
- Backup time

It is advisable to back up instance data during off-peak periods.
- Storage of backup files

Backup files are stored to OBS.
- Handling exceptions in automated backup

If an automated backup task is triggered while the DCS instance is restarting or being scaled up, the backup task will be run in the next cycle.

If backing up a DCS instance fails or the backup is postponed because another task is in progress, DCS will try to back up the instance in the next cycle. A maximum of three retries are allowed within a single day.
- Retention period of backup data

Automated backup files are retained for up to seven days. You can configure the retention period. At the end of the retention period, most backup files of the DCS instance will be automatically deleted, but at least one backup file will be retained.

The latest backup files (up to 24) are always stored unless they are manually deleted.

 **NOTE**

- A total of 24 latest backups (automatic and manual) can be stored. To store the 25th backup, the earliest one will be automatically deleted.
- Deleting an instance removes its backups. To restore them, download and save them in advance.

Data Restoration

- Data restoration process
 - a. You can initiate a data restoration request using the DCS console.
 - b. DCS obtains the backup file from OBS.
 - c. Read/write to the DCS instance is suspended.
 - d. The original data persistence file of the master cache node is replaced by the backup file.
 - e. The new data persistence file (that is, the backup file) is reloaded.
 - f. Data is restored, and the DCS instance starts to provide read/write service again.
- Impact on service systems

Restoration tasks are run on master cache nodes. During restoration, data cannot be written into or read from instances.
- Handling data restoration exceptions

If a backup file is corrupted, DCS will try to fix the backup file while restoring instance data. If the backup file is successfully fixed, the restoration proceeds. If the backup file cannot be fixed, the master/standby DCS instance will be changed back to the state in which it was before data restoration.

7.2 Configuring an Automatic Backup Policy

On the DCS console, you can configure an automatic backup policy. The system then backs up data in your instances according to the backup policy.

By default, automatic backup is disabled. To enable it, perform the operations described in this section. Single-node instances do not support backup and restoration.


If automatic backup is not required, disable the automatic backup function in the backup policy.

Prerequisites

A master/standby, cluster, or read/write splitting DCS instance is in the **Running** state.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Filter DCS instances to find the desired DCS instance. Currently, you can search instances by name, specification, ID, IP address, AZ, status, instance type, cache engine, and many other attributes.

Step 4 Click the name of the desired DCS instance to go to the details page.

Step 5 On the instance details page, click **Backups & Restorations**.


Step 6 Slide  to the right to enable automatic backup. Backup policies will be displayed.

Table 7-1 Parameters in a backup policy

Parameter	Description
Backup Schedule	Day of a week on which data in the chosen DCS instance is automatically backed up. You can select one or multiple days of a week.
Retention Period (days)	The number of days that automatically backed up data is retained. Backup data will be permanently deleted at the end of retention period and cannot be restored. Value range: 1–7.
Start Time	Time at which automatic backup starts. Value: the full hour between 00:00 to 23:00 DCS checks backup policies once every hour. If the backup start time in a backup policy has arrived, data in the corresponding instance is backed up. NOTE Instance backup takes 5 to 30 minutes. The data added or modified during the backup process will not be backed up. To reduce the impact of backup on services, it is recommended that data should be backed up during off-peak periods. Only instances in the Running state can be backed up.

Step 7 Click **OK**.

Step 8 Automatic backup starts at the scheduled time. You can view backup records on the current page.

After the backup is complete, click **Download**, **Restore**, or **Delete** next to the backup record as required.

----End

7.3 Manually Backing Up a DCS Instance

You can manually back up data in DCS instances in a timely manner. This section describes how to manually back up data in master/standby instances using the DCS console.

The latest backup files (up to 24) are always stored unless they are manually deleted.

NOTE


- A total of 24 latest backups (automatic and manual) can be stored. To store the 25th backup, the earliest one will be automatically deleted.
- Deleting an instance removes its backups. To restore them, download and save them in advance.

Prerequisites

A master/standby, cluster, or read/write splitting DCS instance is in the **Running** state.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Filter DCS instances to find the desired DCS instance. Currently, you can search instances by name, specification, ID, IP address, AZ, status, instance type, cache engine, and many other attributes.

Step 4 Click the name of the desired DCS instance to go to the details page.

Step 5 On the instance details page, click **Backups & Restorations**.

Step 6 Click **Create Backup**.

Step 7 Select RDB or AOF for the backup file format.

NOTE

If you select AOF, data will be backed up on the standby node first. The standby node's AOF will be rewritten.

Step 8 In the **Create Backup** dialog box, click **OK**.

Information in the **Description** text box cannot exceed 128 bytes.

 **NOTE**

Instance backup takes 10 to 15 minutes. The data added or modified during the backup process will not be backed up.

----End

7.4 Restoring a DCS Instance

On the DCS console, you can restore backup data to a chosen DCS instance.

 **NOTE**


You can [enable or disable multi-DB](#) for a Proxy Cluster instance. Data backed up when multi-DB is enabled cannot be restored to the instance after multi-DB is disabled.

Prerequisites

- A master/standby, cluster, or read/write splitting DCS instance is in the **Running** state.
- A backup task has been run to back up data in the instance to be restored and the backup task succeeded.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Filter DCS instances to find the desired DCS instance. Currently, you can search instances by name, specification, ID, IP address, AZ, status, instance type, cache engine, and many other attributes.

Step 4 Click the name of the desired DCS instance to go to the details page.

Step 5 On the instance details page, click **Backups & Restorations**.

A list of historical backup tasks is then displayed.

Step 6 Click **Restore** in the row containing the chosen backup task.

Step 7 Click **OK** to start instance restoration.

Information in the **Description** text box cannot exceed 128 bytes.

You can view the results of all restoration tasks on the **Restoration History** page. The records cannot be deleted.

 **NOTE**

Instance restoration takes 1 to 30 minutes.

While being restored, DCS instances do not accept data operation requests from clients because existing data is being overwritten by the backup data.

----End

7.5 Downloading an RDB or AOF Backup File


Automatically backed up data can be retained for a maximum of 7 days. Manually backed up data is not free of charge and takes space in OBS. Due to these limitations, you are advised to download the RDB and AOF backup files and permanently save them on the local host.

This function is supported only by master/standby, read/write splitting, and cluster instances, and not by single-node instances. To export the data of a single-node instance to an RDB file, you can use redis-cli. For details, see [How Do I Export DCS Redis Instance Data?](#)

Prerequisites

The instance has been backed up and the backup is still valid.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
Filter DCS instances to find the desired DCS instance.
- Step 4** Click the name of the DCS instance to display more details about the DCS instance.
- Step 5** On the instance details page, click **Backups & Restorations**.
A list of historical backup tasks is then displayed.
- Step 6** Click **Download** in the row containing the chosen backup task.
- Step 7** In the displayed, **Download Backup File** dialog box, select either of the following two download methods.
Download methods:
 - By URL
 - a. Set the URL validity period and click **Query**.
 - b. Enter URLs in the address bar of the browser or click **Download** to download files.

 NOTE

If you choose to copy URLs, use quotation marks to quote the URLs when running the **wget** command in Linux. For example:

```
wget 'https://obsEndpoint.com:443/redisdemo.rdb?  
parm01=value01&parm02=value02'
```

This is because the URL contains the special character and (&), which will confuse the **wget** command. Quoting the URL facilitates URL identification.

- By OBS
Follow the displayed procedure.

----End

8 Migrating Instance Data

8.1 Data Migration Overview

The DCS console supports online migration (in full or incrementally) and backup migration (by importing backup files) with intuitive operations.

- Backup migration is suitable when the source and target Redis instances are not connected, and the source Redis instance does not support the **SYNC** and **PSYNC** commands. To migrate data, import your backup files to OBS, and DCS will read data from OBS and migrate the data to the target DCS Redis instance. Alternatively, you can import the backup files directly to the DCS instance.
- Online migration is suitable when the source Redis instance supports the **SYNC** and **PSYNC** commands. Data in the source Redis instance can be migrated in full or incrementally to the target instance.

During online migration, the **PSYNC** command is delivered to the source address. For details about how this works, see the [replication explanation](#). This command will cause a fork operation at the source end, which affects latency. For details about the impact scope, see the [Redis official website](#).

 **NOTE**

Currently, the data migration function is free of charge in the OBT. You will be notified when data migration starts to be charged.

For more information about migration tools and schemes, see [Migration Tools and Schemes](#).

Table 8-1 DCS data migration modes

Migration Mode	Source	Target: DCS		
		Single-Node, Read/Write Splitting, or Master/Standby	Proxy Cluster	Redis Cluster

Importing backup files	AOF file	√	√	√
	RDB file	√	√	√
Migrating data online	DCS for Redis: Single-node, read/write splitting, or master/standby	√	√	√
	DCS for Redis: Proxy Cluster	√	√	√
	DCS for Redis: Redis Cluster	√	√	√
	Self-hosted Redis	√	√	√
	Other cloud Redis	√	√	√
<p>NOTE You can migrate data online in full or incrementally from other cloud Redis to DCS for Redis if they are connected and the SYNC and PSYNC commands can be run on the source Redis. However, some instances provided by other cloud vendors may fail to be migrated online. In this case, migrate data through backup import or use other migration schemes. For details, see Migration Tools and Schemes.</p>				

 **NOTE**

- **DCS for Redis** refers to Redis instances provided by DCS.
- **Self-hosted Redis** refers to self-hosted Redis on the cloud, from other cloud vendors, or in on-premises data centers.
- **Other cloud Redis** refers to Redis services provided by other cloud vendors.
- √: Supported. x: Not supported.

8.2 Importing Backup Files from an OBS Bucket

Scenario

Use the DCS console to migrate Redis data from Redis of another cloud or self-hosted Redis to HUAWEI CLOUD DCS for Redis.

Simply download the source Redis data and then upload the data to an OBS bucket in the same region as the target DCS Redis instance. After you have created a migration task on the DCS console, DCS will read data from the OBS bucket and data will be migrated to the target instance.

.aof, .rdb, .zip, and .tar.gz files can be uploaded to OBS buckets. You can directly upload .aof and .rdb files or compress them into .zip or .tar.gz files before uploading.

Prerequisites

- The OBS bucket must be in the same region as the target DCS Redis instance.
- The data files to be uploaded must be in the .aof, .rdb, .zip, or .tar.gz format.
- To migrate data from a single-node or master/standby Redis instance of another cloud, create a backup task and download the backup file.
- To migrate data from a cluster Redis instance of another cloud, download all backup files, upload all of them to the OBS bucket, and select all of them for the migration. Each backup file contains data for a shard of the instance.

Step 1: Prepare the Target DCS Redis Instance

- If a target DCS Redis instance is not available, create one first. For details, see [Buying a DCS Redis Instance](#).
- If you already have a DCS Redis instance, you do not need to create one again, but you need to clear the instance data before the migration. For details, see [Clearing DCS Instance Data](#).
If the instance data is not cleared before the migration and the source and target instances contain the same key, the key in the target instance will be overwritten by the key in the source instance after the migration.
- Redis is backward compatible. The target instance version must be the same as or later than the source instance version.

Step 2: Create an OBS Bucket and Upload Backup Files

Step 1 Upload the backup data files to the OBS bucket by using OBS Browser+.

If the backup file to be uploaded is smaller than 5 GB, go to step [Step 2](#) to upload the file using the OBS console.

If the backup file to be uploaded is larger than 5 GB, follow the [instructions](#) provided by OBS.

Step 2 On the OBS console, upload the backup data files to the OBS bucket.

Perform the following steps if the backup files are smaller than 5 GB:

1. Create an OBS bucket.

When creating an OBS bucket, pay attention to the configuration of the following parameters. For details on how to set other parameters, see [Creating a Bucket](#) in *OBS User Guide*.

- a. **Region:**

The OBS bucket must be in the same region as the target DCS Redis instance.

- b. **Default Storage Class:** Select **Standard** or **Infrequent Access**.

Do not select **Archive**. Otherwise, the migration will fail.

2. In the bucket list, click the bucket created in [Step 2.1](#).
3. In the navigation pane, choose **Objects**.
4. On the **Objects** tab page, click **Upload Object**.
5. Specify **Storage Class**.

Do not select **Archive**. Otherwise, the migration will fail.

6. Upload the objects.
Drag files or folders to the **Upload Object** area or click **add file**.
A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.
 7. Specify **Server-Side Encryption**
 8. Click **Upload**.
- End

Step 3: Create a Migration Task

- Step 1** Log in to the DCS console.
 - Step 2** In the navigation pane, choose **Data Migration**.
 - Step 3** Click **Create Backup Import Task**.
 - Step 4** Enter the task name and description.
 - Step 5** In the **Source Redis** area, select **OBS Bucket** for **Data Source** and then select the OBS bucket to which you have uploaded backup files.
 - Step 6** Click **Add Backup** and select the backup files to be migrated.
 - Step 7** In the **Target Redis** area, select the **Target Redis Instance** prepared in [Step 1: Prepare the Target DCS Redis Instance](#).
 - Step 8** If the target Redis instance has a password, enter the password and click **Test Connection** to check whether the password is correct. If the instance is not password-protected, click **Test Connection** directly.
 - Step 9** Click **Next**.
 - Step 10** Confirm the migration task details and click **Submit**.
Go back to the data migration task list. After the migration is successful, the task status changes to **Successful**.
- End

8.3 Importing Backup Files from Redis

Scenario

Migrate backup data between different DCS Redis instances in the same region. The source instance type must be master/standby, read/write splitting, or cluster.

Simply back up your Redis data, create a migration task on the DCS console, and then import the source backup data to a DCS Redis instance.

Prerequisites

A target DCS Redis instance has been created in the same region as the source Redis. The source instance has data written and has been backed up.

Step 1: Obtain the Source Instance Name

Obtain the name of the source Redis instance.

Step 2: Prepare the Target DCS Redis Instance

- If a DCS Redis instance is not available, create one first. For details, see [Buying a DCS Redis Instance](#).
- If you already have a DCS Redis instance, you do not need to create one again, but you need to clear the instance data before the migration. For details, see [Clearing DCS Instance Data](#).

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

Step 3: Create a Migration Task

Step 1 Log in to the DCS console.

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

Step 3 Click **Create Backup Import Task**.

Step 4 Enter the task name and description.

Step 5 For source Redis, set **Data Source** to **Redis**.

Step 6 For **Source Redis Instance**, select the instance prepared in [Step 1: Obtain the Source Instance Name](#).

Step 7 Select the backup task whose data is to be migrated.

Step 8 For **Target Redis Instance**, select the DCS Redis instance prepared in [Step 2: Prepare the Target DCS Redis Instance](#).

Step 9 If the target Redis instance has a password, enter the password and click **Test Connection** to check whether the password is correct. If the instance is not password-protected, click **Test Connection** directly.

Step 10 Click **Next**.

Step 11 Confirm the migration task details and click **Submit**.

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

----End

8.4 Online Migration

Scenario

If the source and target instances are interconnected and the **SYNC** and **PSYNC** commands are supported by the source instance, data can be migrated online in full or incrementally from the source to the target.

 **CAUTION**

- If the **SYNC** and **PSYNC** commands are disabled on the source Redis instance, enable them before performing online migration. Otherwise, the migration fails. If you use a HUAWEI CLOUD DCS Redis instance for online migration, the **SYNC** command is automatically enabled.
- You cannot use public networks for online migration.
- During online migration, you are advised to set **repl-timeout** on the source instance to 300s and **client-output-buffer-limit** to 20% of the maximum memory of the instance.

 **NOTE**

During online migration, results of the **FLUSHDB** and **FLUSHALL** commands executed on the source will not be synchronized to the target.

Impacts on Services

During online migration, data is essentially synchronized in full to a new replica. Therefore, perform online migration during low-demand hours.

Prerequisites

- Before migrating data, read through [Migration Tools and Schemes](#) to learn about the DCS data migration function and select an appropriate target instance.
- By default, a Proxy Cluster instance has only one database (DB0). Before you migrate data from a single-node, read/write splitting, or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to [Enabling Multi-DB](#).
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a single-node, read/write splitting, or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. If there is any, move all data to DB0 by referring to [Online Migration with Rump](#) to ensure that the migration succeeds.

Step 1: Obtain Information About the Source Redis Instance

- If the source is a DCS Redis instance in the same VPC as the target Redis, obtain the name of the source Redis instance.
- If the source is self-hosted Redis, Redis in another cloud, or a DCS Redis instance in a different VPC from the target Redis, obtain the source IP address or domain name and port number.

Step 2: Prepare the Target DCS Redis Instance

- If a target DCS Redis instance is not available, create one first. For details, see [Buying a DCS Redis Instance](#).
- If you already have a DCS Redis instance, you do not need to create one again, but you need to clear the instance data before the migration. For details, see [Clearing DCS Instance Data](#).

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

Step 3: Check the Network

Step 1 Check whether the source Redis instance, the target Redis instance, and the migration task are configured with the same VPC.

If yes, go to [Step 4: Create an Online Migration Task](#). If no, go to [Step 2](#).

Step 2 Check whether the VPCs configured for the source Redis instance, the target Redis instance, and the migration task are connected to ensure that the VM resource of the migration task can access the source and target Redis instances.

If yes, go to [Step 4: Create an Online Migration Task](#). If no, go to [Step 3](#).

Step 3 Perform the following operations to establish the network.

- If the source and target Redis instances are in the same region, create a VPC peering connection by referring to [VPC Peering Connection](#).
- If the source and target Redis instances are on different clouds, create a connection by referring to [Direct Connect documentation](#).

----End

Step 4: Create an Online Migration Task

Step 1 Log in to the DCS console.

Step 2 In the navigation pane, choose **Data Migration**.

Step 3 Click **Create Online Migration Task**.

Step 4 Enter the task name and description.

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

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

NOTICE

- The migration task uses a tenant IP address (**Migration ECS** displayed on the **Basic Information** page of the task.) If a whitelist is configured for the source or target instance, add the migration IP address to the whitelist or disable the whitelist.
- To allow the VM used by the migration task to access the source and target instances, set an outbound rule for the task's security group to allow traffic through the IP addresses and ports of the source and target instances. By default, all outbound traffic is allowed.

----End

Step 5: Configure the Online Migration Task

Step 1 On the **Online Migration** tab page, click **Configure** in the **Operation** column.

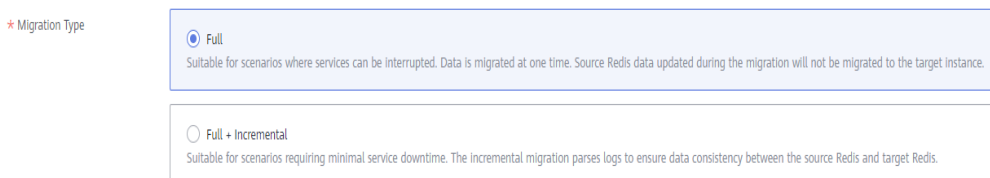
Step 2 Select a migration type.

Supported migration types are **Full** and **Full + Incremental**, which are described in [Table 8-2](#).

Table 8-2 Migration type description

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

Figure 8-1 Selecting the migration type



Step 3 If **Migration Type** is set to **Full + Incremental**, you can specify a bandwidth limit.

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

Step 4 Specify **Auto-Reconnect**. If this option is enabled, automatic reconnections will be performed indefinitely in the case of a network exception.

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

Step 5 Configure source Redis and target Redis.

- The Redis type can be **Redis in the cloud** or **Self-hosted Redis** as required.
 - Redis in the cloud:** a DCS Redis instance (source or target) that is in the same VPC as the migration task. If you select this option, specify a DCS Redis instance.

- **Self-hosted Redis:** a DCS Redis instance, Redis in another cloud, or self-hosted Redis. If you select this option, enter Redis addresses.

NOTE

If the source and target Redis instances are connected but are in different regions of HUAWEI CLOUD, you can only select **Self-hosted Redis** for **Target Redis Type** and enter the instance addresses, regardless of whether the target Redis instance is self-hosted or in the cloud.

2. If the instance is password-protected, click **Test Connection** to check whether the instance password is correct and whether the network is connected. If the instance is not password-protected, click **Test Connection** directly.

Step 6 Click **Next**.

Step 7 Confirm the migration task details and click **Submit**.

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

NOTE

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

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

----End

Verifying the Migration

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

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

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

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

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

8.5 IP Switching

Scenario

Currently, you cannot change the instance type when using the specification modification function. To modify instance specifications while changing the

instance type, you can perform IP switching after data migration. By switching IP addresses, you can also change the AZ and CPU architecture used by an instance.

- After online data migration is complete, you can switch the IP addresses.
- The IP addresses can be rolled back as required after the switching.

Prerequisites

- Obtain information about the source and target instances. For details about preparing a target instance, see [Step 2: Prepare the Target DCS Redis Instance](#).
- Ensure that the source and target instances can communicate with each other. For details, see [Step 3: Check the Network](#).
- The target and source instances must use the same port.
- IP switching can be performed only when the following conditions are met:
 - IP switching depends on the data migration function. Therefore, the source and target instances must support the data migration function. For details, see [Table 8-1](#).
 - Both the source and target instances are Redis instances in the cloud.
 - [Table 8-3](#) lists the supported IP switching scenarios.

Table 8-3 IP switching scenarios

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


Precautions for IP Switching

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

8. Ensure that the passwords of the source and target instances are the same. If they are different, verification will fail after the switching.
9. If a whitelist is configured for the source instance, ensure that the same whitelist is configured for the target instance before switching IP addresses.

Switching IP Addresses

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

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

Step 5 Enter the task name and description.

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

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

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

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

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

NOTE

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


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

----End

Rolling Back IP Addresses

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

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 On the **Online Migration** page, locate the row that contains the migration task in the **IP switched** state, choose **More > Roll Back IP**.

Step 5 In the confirmation dialog box, click **Yes**. The IP address rollback task is submitted successfully. When the task status changes to **IP rolled back**, the rollback is complete.

----End


9 Parameter Templates

9.1 Viewing Parameter Templates

This section describes how to view parameter templates on the DCS console.

Procedure

Step 1 Log in to the DCS console.

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

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

Step 4 Choose the **Default Templates** or **Custom Templates** tab.

Step 5 View parameter templates.

Currently, you can enter a keyword in the search box to search for a parameter template by template name.

Step 6 Click a parameter template. The parameters contained in the template are displayed. For details about the parameters, see [Table 9-1](#).

Table 9-1 DCS Redis instance configuration parameters

Parameter	Description	Value Range	Default Value
timeout	The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of 0 means that this function is disabled. Proxy Cluster instances do not have this parameter.	0-7200 seconds	0
appendfsync	Controls how often fsync() transfers cached data to the disk. Note that some OSs will perform a complete data transfer but some others only make a "best-effort" attempt. Single-node instances do not have this parameter. There are three settings: no: fsync() is never called. The OS will flush data when it is ready. This mode offers the highest performance. always: fsync() is called after every write to the AOF. This mode is very slow, but also very safe. everysec: fsync() is called once per second. This mode provides a compromise between safety and performance.	<ul style="list-style-type: none"> • no • always • everysec 	no

Parameter	Description	Value Range	Default Value
appendonly	<p>Indicates whether to log each modification of the instance. By default, data is written to disks asynchronously in Redis. If this function is disabled, recently-generated data might be lost in the event of a power failure. Single-node instances do not have this parameter.</p> <p>Options:</p> <p>yes: Logs are enabled, that is, persistence is enabled.</p> <p>no: Logs are disabled, that is, persistence is disabled.</p>	<ul style="list-style-type: none"> • yes • no 	yes
client-output-buffer-limit-slave-soft-seconds	Number of seconds that the output buffer remains above client-output-buffer-slave-soft-limit before the client is disconnected.	0-60	60
client-output-buffer-slave-hard-limit	Hard limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected.	0-17,179,869,184	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the soft limit and continuously remains above the limit for the time specified by the client-output-buffer-limit-slave-soft-seconds parameter, the client is disconnected.	0-17,179,869,184	1,717,986,918

Parameter	Description	Value Range	Default Value
maxmemory-policy	<p>The policy applied when the maxmemory limit is reached. Options:</p> <ul style="list-style-type: none"> • volatile-lru: Evict keys by trying to remove the less recently used (LRU) keys first, but only among keys that have an expire set. • allkeys-lru: Evict keys by trying to remove the LRU keys first. • volatile-random: Evict keys randomly, but only among keys that have an expire set. • allkeys-random: Evict keys randomly. • volatile-ttl: Evict keys with an expire set, and try to evict keys with a shorter time to live (TTL) first. • noeviction: Do not delete any keys and only return errors when the memory limit was reached. • volatile-lfu: Evict keys by trying to remove the less frequently used (LFU) keys first, but only among keys that have an expire set. • allkeys-lfu: Evict keys by trying to remove the LFU keys first. <p>For details about eviction policies, see the Redis official website.</p>	<ul style="list-style-type: none"> • volatile-lru • allkeys-lru • volatile-random • allkeys-random • volatile-ttl • noeviction • volatile-lfu • allkeys-lfu 	volatile-lru
lua-time-limit	Maximum time allowed for executing a Lua script (in milliseconds).	100-5000	5000

Parameter	Description	Value Range	Default Value
master-read-only	Sets the instance to be read-only. All write operations will fail. Proxy Cluster instances do not have this parameter.	<ul style="list-style-type: none"> • yes • no 	no
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. This parameter specifies the maximum number of connections on a single node (single shard). <ul style="list-style-type: none"> • Cluster: Maximum connections limit per node = Maximum connections limit of the instance/Shard quantity • Single-node, master/standby, and read/write splitting: Maximum connections limit on a single node = Maximum connections limit of the instance 	1000-50,000	10,000
proto-max-bulk-len	Maximum size of a single element request (in bytes).	1,048,576-536,870,912	536,870,912
repl-backlog-size	The replication backlog size (bytes). The backlog is a buffer that accumulates replica data when replicas are disconnected from the master. When a replica reconnects, a partial synchronization is performed to synchronize the data that was missed while replicas were disconnected.	16,384-1,073,741,824	1,048,576

Parameter	Description	Value Range	Default Value
repl-backlog-ttl	The amount of time, in seconds, before the backlog buffer is released, starting from the last a replica was disconnected. The value 0 indicates that the backlog is never released.	0-604,800	3600
repl-timeout	Replication timeout (in seconds).	30-3600	60
hash-max-ziplist-entries	The maximum number of hashes that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	512
hash-max-ziplist-value	The largest value allowed for a hash encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64
set-max-intset-entries	When a set is composed entirely of strings and number of integer elements is less than this parameter value, the set is encoded using intset, a data structure optimized for memory use.	1-10,000	512
zset-max-ziplist-entries	The maximum number of sorted sets that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	128
zset-max-ziplist-value	The largest value allowed for a sorted set encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64

Parameter	Description	Value Range	Default Value
latency-monitor-threshold	<p>The minimum amount of latency that will be logged as latency spikes</p> <ul style="list-style-type: none"> • Set to 0: Latency monitoring is disabled. • Set to more than 0: All with at least this many ms of latency will be logged. <p>By running the LATENCY command, you can perform operations related to latency monitoring, such as obtaining statistical data, and configuring and enabling latency monitoring.</p> <p>Proxy Cluster instances do not have this parameter.</p>	0-86,400,000 ms	0

Parameter	Description	Value Range	Default Value
notify-keyspace-events	<p>Controls which keyspace events notifications are enabled for. If this parameter is configured, the Redis Pub/Sub feature will allow clients to receive an event notification when a Redis data set is modified.</p> <p>Proxy Cluster instances do not have this parameter.</p>	<p>A combination of different values can be used to enable notifications for multiple event types. Possible values include:</p> <p>K: Keyspace events, published with the <code>__keyspace@*</code> prefix</p> <p>E: Keyevent events, published with <code>__keyevent@*</code> prefix</p> <p>g: Generic commands (non-type specific) such as DEL, EXPIRE, and RENAME</p> <p>\$. String commands</p> <p>l: List commands</p> <p>s: Set commands</p> <p>h: Hash commands</p> <p>z: Sorted set commands</p> <p>x: Expired events (events generated every time a key expires)</p> <p>e: Evicted events (events generated when a key is evicted from maxmemory)</p> <p>For more information, see the following note.</p>	Ex
slowlog-log-slower-than	<p>The maximum amount of time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis slow query log will record the command.</p>	0-1,000,000	10,000

Parameter	Description	Value Range	Default Value
slowlog-max-len	The maximum allowed number of slow queries that can be logged. Slow query log consumes memory, but you can reclaim this memory by running the SLOWLOG RESET command.	0-1000	128
auto-kill-timeout-lua-process	<p>yes: enable no: disable</p> <p>When this parameter is enabled, lua scripts are killed when their execution times out. However, scripts with write operations are not killed, but their nodes automatically restart (if persistence has been enabled for the instance) without saving the write operations.</p> <p>Single-node instances do not have this parameter.</p>	<ul style="list-style-type: none"> • yes • no 	no

 NOTE

1. The default values and value ranges of the **maxclients**, **reserved-memory-percent**, **client-output-buffer-slave-soft-limit**, and **client-output-buffer-slave-hard-limit** parameters are related to the instance specifications. Therefore, these parameters are not displayed in the parameter template.
2. For more information about the parameters described in **Table 9-1**, visit <https://redis.io/topics/memory-optimization>.


----End

9.2 Creating a Custom Parameter Template

You can create custom parameter templates for different cache engine versions and instance types based on service requirements.

Procedure

Step 1 Log in to the DCS console.

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

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

Step 4 Click the **Default Templates** or **Custom Templates** tab to create a template based on a default template or an existing custom template.

- If you select **Default Templates**, click **Customize** in the **Operation** column of the row containing the desired cache engine version.
- If you select **Custom Templates**, click **Copy** in the **Operation** column in the row containing the desired custom template.

Step 5 Specify **Template Name** and **Description**.

 **NOTE**

The template name can contain 4 to 64 characters and must start with a letter or digit. Only letters, digits, hyphens (-), underscores (_), and periods (.) are allowed. The description can be empty.

Step 6 Select **Modifiable parameters**.

Currently, you can enter a keyword in the search box to search for a parameter by parameter name.

Step 7 In the row that contains the parameter to be modified, enter a value in the **Assigned Value** column.

Table 9-2 describes the parameters. In most cases, default values are retained.

Table 9-2 DCS Redis instance configuration parameters

Parameter	Description	Value Range	Default Value
timeout	The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of 0 means that this function is disabled. Proxy Cluster instances do not have this parameter.	0-7200 seconds	0

Parameter	Description	Value Range	Default Value
appendfsync	<p>Controls how often fsync() transfers cached data to the disk. Note that some OSs will perform a complete data transfer but some others only make a "best-effort" attempt. Single-node instances do not have this parameter.</p> <p>There are three settings:</p> <p>no: fsync() is never called. The OS will flush data when it is ready. This mode offers the highest performance.</p> <p>always: fsync() is called after every write to the AOF. This mode is very slow, but also very safe.</p> <p>everysec: fsync() is called once per second. This mode provides a compromise between safety and performance.</p>	<ul style="list-style-type: none"> no always everysec 	no
appendonly	<p>Indicates whether to log each modification of the instance. By default, data is written to disks asynchronously in Redis. If this function is disabled, recently-generated data might be lost in the event of a power failure. Single-node instances do not have this parameter.</p> <p>Options:</p> <p>yes: Logs are enabled, that is, persistence is enabled.</p> <p>no: Logs are disabled, that is, persistence is disabled.</p>	<ul style="list-style-type: none"> yes no 	yes

Parameter	Description	Value Range	Default Value
client-output-buffer-limit-slave-soft-seconds	Number of seconds that the output buffer remains above client-output-buffer-slave-soft-limit before the client is disconnected.	0-60	60
client-output-buffer-slave-hard-limit	Hard limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected.	0-17,179,869,184	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the soft limit and continuously remains above the limit for the time specified by the client-output-buffer-limit-slave-soft-seconds parameter, the client is disconnected.	0-17,179,869,184	1,717,986,918

Parameter	Description	Value Range	Default Value
maxmemory-policy	<p>The policy applied when the maxmemory limit is reached. Options:</p> <ul style="list-style-type: none"> • volatile-lru: Evict keys by trying to remove the less recently used (LRU) keys first, but only among keys that have an expire set. • allkeys-lru: Evict keys by trying to remove the LRU keys first. • volatile-random: Evict keys randomly, but only among keys that have an expire set. • allkeys-random: Evict keys randomly. • volatile-ttl: Evict keys with an expire set, and try to evict keys with a shorter time to live (TTL) first. • noeviction: Do not delete any keys and only return errors when the memory limit was reached. • volatile-lfu: Evict keys by trying to remove the less frequently used (LFU) keys first, but only among keys that have an expire set. • allkeys-lfu: Evict keys by trying to remove the LFU keys first. <p>For details about eviction policies, see the Redis official website.</p>	<ul style="list-style-type: none"> • volatile-lru • allkeys-lru • volatile-random • allkeys-random • volatile-ttl • noeviction • volatile-lfu • allkeys-lfu 	volatile-lru
lua-time-limit	Maximum time allowed for executing a Lua script (in milliseconds).	100-5000	5000

Parameter	Description	Value Range	Default Value
master-read-only	Sets the instance to be read-only. All write operations will fail. Proxy Cluster instances do not have this parameter.	<ul style="list-style-type: none"> • yes • no 	no
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. This parameter specifies the maximum number of connections on a single node (single shard). <ul style="list-style-type: none"> • Cluster: Maximum connections limit per node = Maximum connections limit of the instance/Shard quantity • Single-node, master/standby, and read/write splitting: Maximum connections limit on a single node = Maximum connections limit of the instance 	1000-50,000	10,000
proto-max-bulk-len	Maximum size of a single element request (in bytes).	1,048,576-536,870,912	536,870,912
repl-backlog-size	The replication backlog size (bytes). The backlog is a buffer that accumulates replica data when replicas are disconnected from the master. When a replica reconnects, a partial synchronization is performed to synchronize the data that was missed while replicas were disconnected.	16,384-1,073,741,824	1,048,576

Parameter	Description	Value Range	Default Value
repl-backlog-ttl	The amount of time, in seconds, before the backlog buffer is released, starting from the last a replica was disconnected. The value 0 indicates that the backlog is never released.	0-604,800	3600
repl-timeout	Replication timeout (in seconds).	30-3600	60
hash-max-ziplist-entries	The maximum number of hashes that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	512
hash-max-ziplist-value	The largest value allowed for a hash encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64
set-max-intset-entries	When a set is composed entirely of strings and number of integer elements is less than this parameter value, the set is encoded using intset, a data structure optimized for memory use.	1-10,000	512
zset-max-ziplist-entries	The maximum number of sorted sets that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	128
zset-max-ziplist-value	The largest value allowed for a sorted set encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64

Parameter	Description	Value Range	Default Value
latency-monitor-threshold	<p>The minimum amount of latency that will be logged as latency spikes</p> <ul style="list-style-type: none">• Set to 0: Latency monitoring is disabled.• Set to more than 0: All with at least this many ms of latency will be logged. <p>By running the LATENCY command, you can perform operations related to latency monitoring, such as obtaining statistical data, and configuring and enabling latency monitoring.</p> <p>Proxy Cluster instances do not have this parameter.</p>	0-86,400,000 ms	0

Parameter	Description	Value Range	Default Value
notify-keyspace-events	<p>Controls which keyspace events notifications are enabled for. If this parameter is configured, the Redis Pub/Sub feature will allow clients to receive an event notification when a Redis data set is modified.</p> <p>Proxy Cluster instances do not have this parameter.</p>	<p>A combination of different values can be used to enable notifications for multiple event types. Possible values include:</p> <p>K: Keyspace events, published with the <code>__keyspace@*</code> prefix</p> <p>E: Keyevent events, published with <code>__keyevent@*</code> prefix</p> <p>g: Generic commands (non-type specific) such as DEL, EXPIRE, and RENAME</p> <p>\$. String commands</p> <p>l: List commands</p> <p>s: Set commands</p> <p>h: Hash commands</p> <p>z: Sorted set commands</p> <p>x: Expired events (events generated every time a key expires)</p> <p>e: Evicted events (events generated when a key is evicted from maxmemory)</p> <p>For more information, see the following note.</p>	Ex
slowlog-log-slower-than	<p>The maximum amount of time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis slow query log will record the command.</p>	0-1,000,000	10,000

Parameter	Description	Value Range	Default Value
slowlog-max-len	The maximum allowed number of slow queries that can be logged. Slow query log consumes memory, but you can reclaim this memory by running the SLOWLOG RESET command.	0-1000	128
auto-kill-timeout-lua-process	<p>yes: enable no: disable</p> <p>When this parameter is enabled, lua scripts are killed when their execution times out. However, scripts with write operations are not killed, but their nodes automatically restart (if persistence has been enabled for the instance) without saving the write operations.</p> <p>Single-node instances do not have this parameter.</p>	<ul style="list-style-type: none"> • yes • no 	no

 **NOTE**

1. The default values and value ranges of the **maxclients**, **reserved-memory-percent**, **client-output-buffer-slave-soft-limit**, and **client-output-buffer-slave-hard-limit** parameters are related to the instance specifications. Therefore, these parameters cannot be modified.
2. For more information about the parameters described in [Table 9-2](#), visit <https://redis.io/topics/memory-optimization>.
3. The **latency-monitor-threshold** parameter is usually used for fault location. After locating faults based on the latency information collected, change the value of **latency-monitor-threshold** to **0** to avoid unnecessary latency.
4. More about the **notify-keyspace-events** parameter:
 - The parameter setting must contain at least a **K** or **E**.
 - **A** is an alias for "g\$lshzxe" and cannot be used together with any of the characters in "g\$lshzxe".
 - For example, the value **KI** means that Redis will notify Pub/Sub clients about keyspace events and list commands. The value **AKE** means Redis will notify Pub/Sub clients about all events.

Step 8 Click **OK**.


----End

9.3 Modifying a Custom Parameter Template

You can modify the name, description, and parameters of a custom parameter template based on service requirements.

Procedure

Step 1 Log in to the DCS console.

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

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

Step 4 Choose the **Custom Templates** tab.

Step 5 You can modify a custom parameter template in either of the following ways:

- Click **Edit** in the **Operation** column.
 - a. Change the name or modify the description of a template.
 - b. In the **Parameters** area, select **Modifiable parameters**. In the row that contains the parameter to be modified, enter a value in the **Assigned Value** column. [Table 9-3](#) describes the parameters. In most cases, default values are retained.
 - c. Click **OK**.
- Click the name of a custom template. On the displayed page, modify parameters.
 - a. Select **Modifiable parameters**. Enter a keyword in the search box to search for a parameter by parameter name.
 - b. Click **Modify**.
 - c. In the row that contains the parameter to be modified, enter a value in the **Assigned Value** column. [Table 9-3](#) describes the parameters. In most cases, default values are retained.
 - d. Click **Save**.

Table 9-3 DCS Redis instance configuration parameters

Parameter	Description	Value Range	Default Value
timeout	The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of 0 means that this function is disabled. Proxy Cluster instances do not have this parameter.	0-7200 seconds	0
appendfsync	Controls how often fsync() transfers cached data to the disk. Note that some OSs will perform a complete data transfer but some others only make a "best-effort" attempt. Single-node instances do not have this parameter. There are three settings: no: fsync() is never called. The OS will flush data when it is ready. This mode offers the highest performance. always: fsync() is called after every write to the AOF. This mode is very slow, but also very safe. everysec: fsync() is called once per second. This mode provides a compromise between safety and performance.	<ul style="list-style-type: none"> • no • always • everysec 	no

Parameter	Description	Value Range	Default Value
appendonly	<p>Indicates whether to log each modification of the instance. By default, data is written to disks asynchronously in Redis. If this function is disabled, recently-generated data might be lost in the event of a power failure. Single-node instances do not have this parameter.</p> <p>Options:</p> <p>yes: Logs are enabled, that is, persistence is enabled.</p> <p>no: Logs are disabled, that is, persistence is disabled.</p>	<ul style="list-style-type: none"> • yes • no 	yes
client-output-buffer-limit-slave-soft-seconds	Number of seconds that the output buffer remains above client-output-buffer-slave-soft-limit before the client is disconnected.	0-60	60
client-output-buffer-slave-hard-limit	Hard limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected.	0-17,179,869,184	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the soft limit and continuously remains above the limit for the time specified by the client-output-buffer-limit-slave-soft-seconds parameter, the client is disconnected.	0-17,179,869,184	1,717,986,918

Parameter	Description	Value Range	Default Value
maxmemory-policy	<p>The policy applied when the maxmemory limit is reached. Options:</p> <ul style="list-style-type: none"> • volatile-lru: Evict keys by trying to remove the less recently used (LRU) keys first, but only among keys that have an expire set. • allkeys-lru: Evict keys by trying to remove the LRU keys first. • volatile-random: Evict keys randomly, but only among keys that have an expire set. • allkeys-random: Evict keys randomly. • volatile-ttl: Evict keys with an expire set, and try to evict keys with a shorter time to live (TTL) first. • noeviction: Do not delete any keys and only return errors when the memory limit was reached. • volatile-lfu: Evict keys by trying to remove the less frequently used (LFU) keys first, but only among keys that have an expire set. • allkeys-lfu: Evict keys by trying to remove the LFU keys first. <p>For details about eviction policies, see the Redis official website.</p>	<ul style="list-style-type: none"> • volatile-lru • allkeys-lru • volatile-random • allkeys-random • volatile-ttl • noeviction • volatile-lfu • allkeys-lfu 	volatile-lru
lua-time-limit	Maximum time allowed for executing a Lua script (in milliseconds).	100-5000	5000

Parameter	Description	Value Range	Default Value
master-read-only	Sets the instance to be read-only. All write operations will fail. Proxy Cluster instances do not have this parameter.	<ul style="list-style-type: none"> • yes • no 	no
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. This parameter specifies the maximum number of connections on a single node (single shard). <ul style="list-style-type: none"> • Cluster: Maximum connections limit per node = Maximum connections limit of the instance/Shard quantity • Single-node, master/standby, and read/write splitting: Maximum connections limit on a single node = Maximum connections limit of the instance 	1000-50,000	10,000
proto-max-bulk-len	Maximum size of a single element request (in bytes).	1,048,576-536,870,912	536,870,912
repl-backlog-size	The replication backlog size (bytes). The backlog is a buffer that accumulates replica data when replicas are disconnected from the master. When a replica reconnects, a partial synchronization is performed to synchronize the data that was missed while replicas were disconnected.	16,384-1,073,741,824	1,048,576

Parameter	Description	Value Range	Default Value
repl-backlog-ttl	The amount of time, in seconds, before the backlog buffer is released, starting from the last a replica was disconnected. The value 0 indicates that the backlog is never released.	0-604,800	3600
repl-timeout	Replication timeout (in seconds).	30-3600	60
hash-max-ziplist-entries	The maximum number of hashes that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	512
hash-max-ziplist-value	The largest value allowed for a hash encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64
set-max-intset-entries	When a set is composed entirely of strings and number of integer elements is less than this parameter value, the set is encoded using intset, a data structure optimized for memory use.	1-10,000	512
zset-max-ziplist-entries	The maximum number of sorted sets that can be encoded using ziplist, a data structure optimized to reduce memory use.	1-10,000	128
zset-max-ziplist-value	The largest value allowed for a sorted set encoded using ziplist, a special data structure optimized for memory use.	1-10,000	64

Parameter	Description	Value Range	Default Value
latency-monitor-threshold	<p>The minimum amount of latency that will be logged as latency spikes</p> <ul style="list-style-type: none"> • Set to 0: Latency monitoring is disabled. • Set to more than 0: All with at least this many ms of latency will be logged. <p>By running the LATENCY command, you can perform operations related to latency monitoring, such as obtaining statistical data, and configuring and enabling latency monitoring.</p> <p>Proxy Cluster instances do not have this parameter.</p>	0-86,400,000 ms	0

Parameter	Description	Value Range	Default Value
notify-keyspace-events	<p>Controls which keyspace events notifications are enabled for. If this parameter is configured, the Redis Pub/Sub feature will allow clients to receive an event notification when a Redis data set is modified.</p> <p>Proxy Cluster instances do not have this parameter.</p>	<p>A combination of different values can be used to enable notifications for multiple event types. Possible values include:</p> <p>K: Keyspace events, published with the <code>__keyspace@*</code> prefix</p> <p>E: Keyevent events, published with <code>__keyevent@*</code> prefix</p> <p>g: Generic commands (non-type specific) such as DEL, EXPIRE, and RENAME</p> <p>\$. String commands</p> <p>l: List commands</p> <p>s: Set commands</p> <p>h: Hash commands</p> <p>z: Sorted set commands</p> <p>x: Expired events (events generated every time a key expires)</p> <p>e: Evicted events (events generated when a key is evicted from maxmemory)</p> <p>For more information, see the following note.</p>	Ex
slowlog-log-slower-than	<p>The maximum amount of time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis slow query log will record the command.</p>	0-1,000,000	10,000

Parameter	Description	Value Range	Default Value
slowlog-max-len	The maximum allowed number of slow queries that can be logged. Slow query log consumes memory, but you can reclaim this memory by running the SLOWLOG RESET command.	0-1000	128
auto-kill-timeout-lua-process	<p>yes: enable no: disable</p> <p>When this parameter is enabled, lua scripts are killed when their execution times out. However, scripts with write operations are not killed, but their nodes automatically restart (if persistence has been enabled for the instance) without saving the write operations.</p> <p>Single-node instances do not have this parameter.</p>	<ul style="list-style-type: none"> • yes • no 	no

 **NOTE**


1. The default values and value ranges of the **maxclients**, **reserved-memory-percent**, **client-output-buffer-slave-soft-limit**, and **client-output-buffer-slave-hard-limit** parameters are related to the instance specifications. Therefore, these parameters cannot be modified.
2. For more information about the parameters described in [Table 9-3](#), visit <https://redis.io/topics/memory-optimization>.
3. The **latency-monitor-threshold** parameter is usually used for fault location. After locating faults based on the latency information collected, change the value of **latency-monitor-threshold** to **0** to avoid unnecessary latency.
4. More about the **notify-keyspace-events** parameter:
 - The parameter setting must contain at least a **K** or **E**.
 - **A** is an alias for "g\$lshzxe" and cannot be used together with any of the characters in "g\$lshzxe".
 - For example, the value **KI** means that Redis will notify Pub/Sub clients about keyspace events and list commands. The value **AKE** means Redis will notify Pub/Sub clients about all events.

----End

9.4 Deleting a Custom Parameter Template

This section describes how to delete a custom parameter template.

Procedure

- Step 1** Log in to the DCS console.
 - Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
 - Step 3** In the navigation pane, choose **Parameter Templates**.
 - Step 4** Choose the **Custom Templates** tab.
 - Step 5** Click **Delete** in the **Operation** column.
 - Step 6** Click **Yes**.
- End

10 Managing Passwords

10.1 DCS Instance Passwords

Passwords can be configured to control access to your DCS instances, ensuring the security of your data.

You can set a password during or after instance creation. For details on how to set a password after an instance has been created, see [Resetting Instance Passwords](#).

You can choose whether to enable password-free access based on your security and convenience trade-off.

Scenarios Requiring Passwords

- For a DCS instance that is used on the live network or contains important information, you are advised to set a password.
- For a DCS instance with public access enabled, a password must be set to ensure data security.

For details on how to access an instance with a password, see [Accessing a DCS Instance](#).

Using Passwords Securely

1. Hide the password when using redis-cli.

If the **-a <password>** option is used in redis-cli in Linux, the password is prone to leakage because it is logged and kept in the history. You are advised not to use the **-a <password>** option when running commands in redis-cli. After you have connected to Redis, run the **auth** command to complete authentication, as shown in the following example:

```
$ redis-cli -h 192.168.0.148 -p 6379
redis 192.168.0.148:6379>auth yourPassword
OK
redis 192.168.0.148:6379>
```

2. Use interactive password authentication or switch between users with different permissions.

If the script involves DCS instance access, use interactive password authentication. To enable automatic script execution, manage the script as another user and authorize execution using `sudo`.

3. Use an encryption module in your application to encrypt the password.

10.2 Changing Instance Passwords

On the DCS console, you can change the password required for accessing your DCS instance.


NOTE

- You cannot change the password of a DCS instance in password-free mode.
- The DCS instance for which you want to change the password is in the **Running** state.
- The new password takes effect immediately on the server without requiring a restart. The client must reconnect to the server using the new password after a `pconnect` connection is closed. (The old password can still be used before disconnection.)

Prerequisites

A DCS instance has been created.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Choose **More > Change Password** in the row containing the chosen instance.
- Step 5** In the displayed dialog box, set **Old Password**, **New Password**, and **Confirm Password**.

NOTE

After 5 consecutive incorrect password attempts, the account for accessing the chosen DCS instance will be locked for 5 minutes. Passwords cannot be changed during the lockout period.

The password must meet the following requirements:

- Cannot be left blank.
- Cannot be the same as the old password.
- Can be 8 to 64 characters long.
- Contain at least three of the following character types:
 - Lowercase letters
 - Uppercase letters
 - Digits
 - special characters (`~!@#%^&*()-_+=\|{}<.>/?`)

Step 6 In the **Change Password** dialog box, click **OK** to confirm the password change.

----End

10.3 Resetting Instance Passwords

On the DCS console, you can configure a new password if you forget your instance password.

NOTE


- You can change it from password mode to password-free mode or from password-free mode to password mode by resetting its password. For details, see [Changing Password Settings for DCS Redis Instances](#).
- The DCS instance for which you want to reset the password is in the **Running** state.
- The new password takes effect immediately on the server without requiring a restart. The client must reconnect to the server using the new password after a pconnect connection is closed. (The old password can still be used before disconnection.)

Prerequisites

A DCS instance has been created.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 Choose **More > Reset Password** in the row containing the chosen instance.

Step 5 In the displayed dialog box, set **New Password**, and **Confirm Password**.

NOTE

The password must meet the following requirements:

- Cannot be left blank.
- Can be 8 to 64 characters long.
- Contain at least three of the following character types:
 - Lowercase letters
 - Uppercase letters
 - Digits
 - special characters (`~!@#$%^&*()-_+=\|{}<.>/?`)

Step 6 Click **OK**.

 **NOTE**

The system will display a success message only after the password is successfully reset on all nodes. If the reset fails, the instance will restart and the password of the cache instance will be restored.

----End

10.4 Changing Password Settings for DCS Redis Instances

Scenario

DCS Redis instances can be accessed with or without passwords. After an instance is created, you can change its password setting:


- To access a DCS Redis instance in password-free mode, you can enable password-free access to clear the existing password of the instance.

 **NOTE**

- To change the password setting, the DCS Redis instance must be in the **Running** state.
- Password-free access may compromise security. You can set a password by using the password reset function.
- For security purposes, password-free access must be disabled when public access is enabled.

Procedure

Step 1 Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.

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

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

Step 4 To change the password setting for a DCS Redis instance, choose **Operation > More > Reset Password** in the row containing the chosen instance.

Step 5 In the **Reset Password** dialogue box, perform either of the following operations as required:

- From password-protected to password-free:
Switch the toggle for **Password-Free Access** and click **OK**.
- From password-free to password-protected:
Enter a password, confirm the password, and click **OK**.

----End

11 Quotas

What Is Quota?

A quota is a limit on the quantity or capacity of a certain type of service resources that you can use, for example, the maximum number of DCS instances that you can create and the maximum amount of memory that you can use.

If a quota cannot meet your needs, apply for a higher quota.

How Do I View My Quota?


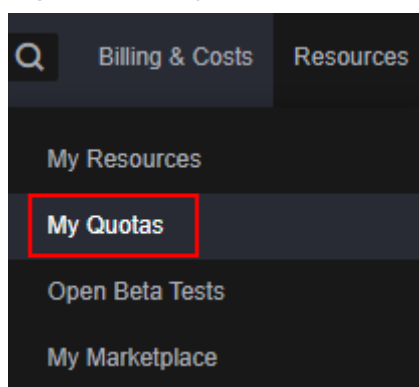
1. Log in to the management console.
2. Click  in the upper left corner of the management console and select the region where your instance is located.
3. In the upper right corner of the page, choose **Resources > My Quotas**. The **Service Quota** page is displayed.

Figure 11-1 My Quotas

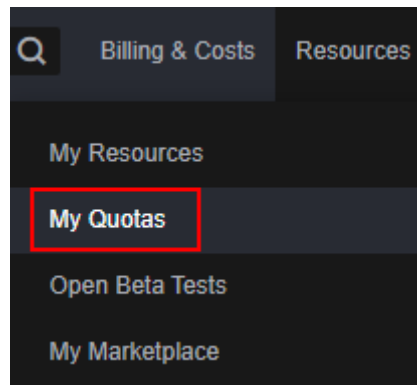


4. On the **Service Quota** page, view the used and total quotas of resources. If a quota cannot meet your needs, apply for a higher quota by performing the following operations.

How Do I Increase My Quota?

1. Log in to the management console.
2. In the upper right corner of the page, choose **Resources > My Quotas**.
The **Service Quota** page is displayed.

Figure 11-2 My Quotas



3. Click **Increase Quota**.
4. On the **Create Service Ticket** page, set the parameters.
In the **Problem Description** area, enter the required quota and the reason for the quota adjustment.
5. Read the agreements and confirm that you agree to them, and then click **Submit**.

12 Monitoring

Cloud Eye is a secure, scalable monitoring platform. It monitors DCS metrics, and sends notifications if alarms are triggered or events occur.

12.1 DCS Metrics

Introduction

This section describes DCS metrics reported to Cloud Eye as well as their namespaces and dimensions. You can use the Cloud Eye console or call [APIs](#) to query the DCS metrics and alarms.

Different types of instances are monitored on different dimensions.

Table 12-1 Monitoring dimensions for different instance types

Instance Type	Instance Monitoring	Redis Server Monitoring	Proxy Monitoring
Single-node	Supported The monitoring on the instance dimension is conducted on the Redis Server.	N/A	N/A
Master/standby	Supported The master node is monitored.	Supported The master and standby nodes are monitored.	N/A
Read/write splitting	Supported The master node is monitored.	Supported The master and standby nodes are monitored.	Supported Each proxy is monitored.

Instance Type	Instance Monitoring	Redis Server Monitoring	Proxy Monitoring
Proxy Cluster	Supported The monitoring data is the aggregated master node data.	Supported Each shard is monitored.	Supported Each proxy is monitored.
Redis Cluster	Supported The monitoring data is the aggregated master node data.	Supported Each shard is monitored.	N/A

Namespace

SYS.DCS

DCS Redis 4.0/5.0/6.0 Instance Metrics

NOTE

- [Dimensions](#) lists the metric dimensions.
- The monitoring data is the aggregated master node data.
- Some metrics are aggregated from the master and replica nodes. For details, see "Metric Description" in [Table 12-2](#).

Table 12-2 DCS Redis 4.0/5.0/6.0 instance metrics

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
cpu_usage	CPU Usage	The monitored object's maximum CPU usage among multiple sampling values in a monitoring period Unit: %	0–100%	Single-node, master/standby, or read/write splitting DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
cpu_avg_usage	Average CPU Usage	The monitored object's average CPU usage of multiple sampling values in a monitoring period Unit: %	0–100%	Single-node, master/standby, or read/write splitting DCS Redis instance	1 minute
command_max_delay	Maximum Command Latency	Maximum latency of commands Unit: ms	≥ 0 ms	DCS Redis instance	1 minute
total_connections_received	New Connections	Number of connections received during the monitoring period	≥ 0	DCS Redis instance	1 minute
is_slow_log_exist	Slow Query Logs	Existence of slow query logs in the instance NOTE Slow queries caused by the MIGRATE , SLAVEOF , CONFIG , BGSAVE , and BGREWRITEAOF commands are not counted.	<ul style="list-style-type: none"> • 1: yes • 0: no 	DCS Redis instance	1 minute
memory_usage	Memory Usage	Memory consumed by the monitored object Unit: %	0–100%	DCS Redis instance	1 minute
expires	Keys With an Expiration	Number of keys with an expiration in Redis	≥ 0	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
keyspace_hits_perc	Hit Rate	Ratio of the number of Redis cache hits to the number of lookups. Calculation: $\text{keyspace_hits} / (\text{keyspace_hits} + \text{keyspace_misses})$ Aggregated from the master and replica nodes. Unit: %	0–100%	DCS Redis instance	1 minute
used_memory	Used Memory	Total number of bytes used by the Redis server Unit: KB, MB, or byte (configurable on the console)	≥ 0	DCS Redis instance	1 minute
used_memory_dataset	Used Memory Dataset	Dataset memory that the Redis server has used Unit: KB, MB, or byte (configurable on the console)	≥ 0	DCS Redis instance	1 minute
used_memory_dataset_perc	Used Memory Dataset Ratio	Percentage of dataset memory that server has used Aggregated from the master and replica nodes. Unit: %	0–100%	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
used_memory_rss	Used Memory RSS	Resident set size (RSS) memory that the Redis server has used, which is the memory that actually resides in the memory, including all stack and heap memory but not swapped-out memory Unit: KB, MB, or byte (configurable on the console)	≥ 0	DCS Redis instance	1 minute
instantaneous_ops	Ops per Second	Number of commands processed per second	≥ 0	DCS Redis instance	1 minute
keyspace_misses	Keyspace Misses	Number of failed lookups of keys in the main dictionary during the monitoring period Aggregated from the master and replica nodes.	≥ 0	DCS Redis instance	1 minute
keys	Keys	Number of keys in Redis	≥ 0	DCS Redis instance	1 minute
blocked_clients	Blocked Clients	Number of clients suspended by block operations	≥ 0	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
connected_clients	Connected Clients	Number of connected clients. Includes connections established for system monitoring, configuration synchronization, and services. Excludes connections from replicas.	≥ 0	DCS Redis instance	1 minute
del	DEL	Number of DEL commands processed per second	0–500,000	DCS Redis instance	1 minute
evicted_keys	Evicted Keys	Number of keys that have been evicted and deleted during the monitoring period Aggregated from the master and replica nodes.	≥ 0	DCS Redis instance	1 minute
expire	EXPIRE	Number of EXPIRE commands processed per second	0–500,000	DCS Redis instance	1 minute
expired_keys	Expired Keys	Number of keys that have expired and been deleted during the monitoring period Aggregated from the master and replica nodes.	≥ 0	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
get	GET	Number of GET commands processed per second Aggregated from the master and replica nodes. Unit: count/s	0–500,000	DCS Redis instance	1 minute
hdel	HDEL	Number of HDEL commands processed per second	0–500,000	DCS Redis instance	1 minute
hget	HGET	Number of HGET commands processed per second Aggregated from the master and replica nodes. Unit: count/s	0–500,000	DCS Redis instance	1 minute
hmget	HMGET	Number of HMGET commands processed per second Aggregated from the master and replica nodes. Unit: count/s	0–500,000	DCS Redis instance	1 minute
hmset	HMSET	Number of HMSET commands processed per second	0–500,000	DCS Redis instance	1 minute
hset	HSET	Number of HSET commands processed per second	0–500,000	DCS Redis instance	1 minute
instantaneous_input_kbps	Input Flow	Instantaneous input traffic Unit: KB/s	≥ 0 KB/s	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
instantaneous_output_kbps	Output Flow	Instantaneous output traffic Unit: KB/s	≥ 0 KB/s	DCS Redis instance	1 minute
memory_frag_ratio	Memory Fragmentation Ratio	Ratio between Used Memory RSS and Used Memory	≥ 0	DCS Redis instance	1 minute
mget	MGET	Number of MGET commands processed per second Aggregated from the master and replica nodes. Unit: count/s	0–500,000	DCS Redis instance	1 minute
mset	MSET	Number of MSET commands processed per second	0–500,000	DCS Redis instance	1 minute
pubsub_channels	PubSub Channels	Number of Pub/Sub channels	≥ 0	DCS Redis instance	1 minute
pubsub_patterns	PubSub Patterns	Number of Pub/Sub patterns	≥ 0	DCS Redis instance	1 minute
set	SET	Number of SET commands processed per second	0–500,000	DCS Redis instance	1 minute
used_memory_lua	Used Memory Lua	Number of bytes used by the Lua engine Unit: KB, MB, or byte (configurable on the console)	≥ 0	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
used_memory_peak	Used Memory Peak	Peak memory consumed by Redis since the Redis server last started Unit: KB, MB, or byte (configurable on the console)	≥ 0	DCS Redis instance	1 minute
sadd	Sadd	Number of SADD commands processed per second Unit: count/s	0–500,000	DCS Redis instance	1 minute
smembers	Smembers	Number of SMEMBERS commands processed per second Aggregated from the master and replica nodes. Unit: count/s	0–500,000	DCS Redis instance	1 minute
scan	SCAN	Number of SCAN operations per second Unit: count/s	0–500,000	DCS Redis instance	1 minute
setex	SETEX	Number of SETEX operations per second Unit: count/s	0–500,000	DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
rx_controlled	Flow Control Times	Number of flow control times during the monitoring period If the value is greater than 0, the used bandwidth exceeds the upper limit and flow control is triggered. Unit: Count	≥ 0	DCS Redis instance	1 minute
bandwidth_usage	Bandwidth Usage	Percentage of the used bandwidth to the maximum bandwidth limit	0–200%	DCS Redis instance	1 minute
command_max_rt	Maximum Latency	Maximum delay from when the node receives commands to when it responds Unit: us	≥ 0	Single-node DCS Redis instance	1 minute
command_avg_rt	Average Latency	Average delay from when the node receives commands to when it responds Unit: us	≥ 0	Single-node DCS Redis instance	1 minute

Redis Server Metrics of DCS Redis Instances

NOTE

- These metrics are supported for master/standby, read/write splitting, and cluster instances.
- [Dimensions](#) lists the metric dimensions.

Table 12-3 Redis Server metrics

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
cpu_usage	CPU Usage	The monitored object's maximum CPU usage among multiple sampling values in a monitoring period Unit: %	0–100%	Redis Server of a master/standby, read/write splitting, or cluster DCS Redis instance	1 minute
cpu_avg_usage	Average CPU Usage	The monitored object's average CPU usage of multiple sampling values in a monitoring period Unit: %	0–100%	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
memory_usage	Memory Usage	Memory consumed by the monitored object Unit: %	0–100%	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
connected_clients	Connected Clients	Number of connected clients. Includes connections established for system monitoring, configuration synchronization, and services. Excludes connections from replicas	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
client_longest_output_list	Client Longest Output List	Longest output list among current client connections	≥ 0	Redis 4.0 Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
client_biggest_in_buf	Client Biggest Input Buf	Maximum input data length among current client connections Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis 4.0 Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
blocked_clients	Blocked Clients	Number of clients suspended by block operations such as BLPOP, BRPOP, and BRPOPLPUSH	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
used_memory	Used Memory	Total number of bytes used by the Redis server Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
used_memory_rss	Used Memory RSS	RSS memory that the Redis server has used, which includes all stack and heap memory but not swapped-out memory Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
used_memory_peak	Used Memory Peak	Peak memory consumed by Redis since the Redis server last started Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
used_memory_lua	Used Memory Lua	Number of bytes used by the Lua engine Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
memory_fragmentation_ratio	Memory Fragmentation Ratio	Current memory fragmentation, which is the ratio between used_memory_rss/used_memory .	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
total_connections_received	New Connections	Number of connections received during the monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
total_commands_processed	Commands Processed	Number of commands processed during the monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
instantaneous_ops_per_sec	Ops per Second	Number of commands processed per second	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
total_net_input_bytes	Network Input Bytes	Number of bytes received during the monitoring period Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
total_net_output_bytes	Network Output Bytes	Number of bytes sent during the monitoring period Unit: KB, MB, or byte (configurable on the console)	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
instantaneous_input_kbps	Input Flow	Instantaneous input traffic Unit: KB/s	≥ 0 KB/s	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
instantaneous_output_kbps	Output Flow	Instantaneous output traffic Unit: KB/s	≥ 0 KB/s	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
rejected_connections	Rejected Connections	Number of connections that have exceeded maxclients and been rejected during the monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
expired_keys	Expired Keys	Number of keys that have expired and been deleted during the monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
evicted_keys	Evicted Keys	Number of keys that have been evicted and deleted during the monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
pubsub_channels	PubSub Channels	Number of Pub/Sub channels	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
pubsub_patterns	PubSub Patterns	Number of Pub/Sub patterns	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
keyspace_hits_perc	Hit Rate	Ratio of the number of Redis cache hits to the number of lookups. Calculation: $\text{keyspace_hits} / (\text{keyspace_hits} + \text{keyspace_misses})$ Unit: %	0–100%	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
command_max_delay	Maximum Command Latency	Maximum latency of commands Unit: ms	≥ 0 ms	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
is_slow_log_exist	Slow Query Logs	Existence of slow query logs in the node NOTE Slow queries caused by the MIGRATE , SLAVEOF , CONFIG , BGSAVE , and BGREWRITEAOF commands are not counted.	<ul style="list-style-type: none"> ● 1: yes ● 0: no 	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
keys	Keys	Number of keys in Redis	≥ 0	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
sadd	SADD	Number of SADD commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
smembers	SMEMBERS	Number of SMEMBERS commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
ms_repl_offset	Replication Gap	Data synchronization gap between the master and the replica	-	Replica of a master/standby, read/write splitting, or cluster instance	1 minute
del	DEL	Number of DEL commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
expire	EXPIRE	Number of EXPIRE commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
get	GET	Number of GET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
hdel	HDEL	Number of HDEL commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
hget	HGET	Number of HGET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
hmget	HMGET	Number of HMGET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
hmset	HMSET	Number of HMSET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby or cluster instance	1 minute
hset	HSET	Number of HSET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
mget	MGET	Number of MGET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
mset	MSET	Number of MSET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
set	SET	Number of SET commands processed per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
rx_controlled	Flow Control Times	Number of flow control times during the monitoring period If the value is greater than 0, the used bandwidth exceeds the upper limit and flow control is triggered. Unit: Count	≥ 0	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
bandwidth_usage	Bandwidth Usage	Percentage of the used bandwidth to the maximum bandwidth limit	0–200%	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
connections_usage	Connection Usage	Percentage of the current number of connections to the maximum allowed number of connections Unit: %	0–100%	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
command_max_rt	Maximum Latency	Maximum delay from when the node receives commands to when it responds Unit: us	≥ 0	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
command_avg_rt	Average Latency	Average delay from when the node receives commands to when it responds Unit: us	≥ 0	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
sync_full	Full Sync Times	Total number of full synchronizations since the Redis Server last started	≥ 0	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
slow_log_counts	Slow Queries	Number of times that slow queries occur within a monitoring period	≥ 0	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute
scan	SCAN	Number of SCAN operations per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
setex	SETEX	Number of SETEX operations per second Unit: count/s	0–500,000	Redis Server of a master/standby, read/write splitting, or cluster instance	1 minute

Proxy Metrics

NOTE

- These metrics are supported by Proxy Cluster and read/write splitting instances.
- [Dimensions](#) lists the metric dimensions.

Table 12-4 Proxy metrics of Proxy Cluster or read/write splitting DCS Redis 4.0 or 5.0 instances

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
node_status	Proxy Status	Indication of whether the proxy is normal.	<ul style="list-style-type: none"> • 0: Normal • 1: Abnormal 	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
cpu_usage	CPU Usage	The monitored object's maximum CPU usage among multiple sampling values in a monitoring period Unit: %	0–100%	Proxy in a Proxy Cluster or read/write splitting instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
cpu_avg_usage	Average CPU Usage	The monitored object's average CPU usage of multiple sampling values in a monitoring period Unit: %	0-100%	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
memory_usage	Memory Usage	Memory consumed by the monitored object Unit: %	0-100%	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
connected_clients	Connected Clients	Number of connected clients. Includes connections established for system monitoring, configuration synchronization, and services. Excludes connections from replicas	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
instantaneous_ops	Ops per Second	Number of commands processed per second	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
instantaneous_input_kbps	Input Flow	Instantaneous input traffic Unit: KB/s	≥ 0 KB/s	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
instantaneous_output_kbps	Output Flow	Instantaneous output traffic Unit: KB/s	≥ 0 KB/s	Proxy in a Proxy Cluster or read/write splitting instance	1 minute

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
total_net_input_bytes	Network Input Bytes	Number of bytes received during the monitoring period Unit: KB, MB, or byte (configurable on the console)	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
total_net_output_bytes	Network Output Bytes	Number of bytes sent during the monitoring period Unit: KB, MB, or byte (configurable on the console)	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
connections_usage	Connection Usage	Percentage of the current number of connections to the maximum allowed number of connections Unit: %	0-100%	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
command_max_rt	Maximum Latency	Maximum delay from when the node receives commands to when it responds Unit: us	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute
command_avg_rt	Average Latency	Average delay from when the node receives commands to when it responds Unit: us	≥ 0	Proxy in a Proxy Cluster or read/write splitting instance	1 minute

Dimensions

Key	Value
dcs_instance_id	DCS Redis instance

Key	Value
dcx_cluster_redis_node	Redis Server
dcx_cluster_proxy2_node	Proxy in a Proxy Cluster and read/write splitting DCS Redis 4.0 or 5.0 instance

12.2 Common Metrics

This section describes common Redis metrics.

Table 12-5 Common metrics


Metric	Description
CPU Usage	<p>This metric indicates the maximum value in each measurement period (minute-level: every minute; second-level: every 5 seconds).</p> <ul style="list-style-type: none"> For a single-node or master/standby instance, you can view the CPU usage of the instance. For a Proxy Cluster instance, you can view the CPU usage of the Redis Servers and the proxies. For a Redis Cluster instance, you can only view the CPU usage of the Redis Servers.
Memory Usage	<p>This metric measures the memory usage in each measurement period (minute-level: every minute; second-level: every 5 seconds).</p> <ul style="list-style-type: none"> For a single-node or master/standby instance, you can view the memory usage of the instance. For a Proxy Cluster instance, you can view the memory usage of the instance and the proxies. For a Redis Cluster instance, you can only view the memory usage of the Redis Servers. <p>NOTICE The memory usage does not include the usage of reserved memory.</p>
Connected Clients	<p>This metric indicates the number of instantaneous connected clients, that is, the number of concurrent connections.</p> <p>This metric does not include the number of connections to the standby nodes of master/standby or cluster instances.</p> <p>For details about the maximum allowed number of connections, see the "Max. Connections" column of different instance types listed in DCS Instance Specifications.</p>

Metric	Description
Ops per Second	<p>This metric indicates the number of operations processed per second.</p> <p>For details about the maximum allowed number of operations per second, see the "Reference Performance (QPS)" column of different instance types listed in DCS Instance Specifications.</p>
Input Flow	<p>This metric indicates the instantaneous input traffic.</p> <ul style="list-style-type: none"> • The monitoring data on the instance level shows the aggregated input traffic of all nodes. • The monitoring data on the node level shows the input traffic of the current node.
Output Flow	<p>This metric indicates the instantaneous output traffic.</p> <ul style="list-style-type: none"> • The monitoring data on the instance level shows the aggregated output traffic of all nodes. • The monitoring data on the node level shows the output traffic of the current node.
Bandwidth Usage	<p>This metric indicates the percentage of the used bandwidth to the maximum bandwidth limit.</p> <p>Bandwidth usage = (Input flow + Output flow)/(2 x Maximum bandwidth) x 100%</p>
Commands Processed	<p>This metric indicates the number of commands processed during the monitoring period. The default monitoring period is 1 minute.</p> <p>The monitoring period of this metric is different from that of the Ops per Second metric. The Ops per Second metric measures the instantaneous number of commands processed. The Commands Processed metric measures the total number of commands processed during the monitoring period.</p>
Flow Control Times	<p>This metric indicates the number of times that the maximum allowed bandwidth is exceeded during the monitoring period.</p> <p>For details about the maximum allowed bandwidth, see the "Maximum/Assured Bandwidth" column of different instance types listed in DCS Instance Specifications.</p>
Slow Queries	<p>This metric indicates whether slow queries exist on the instance.</p> <p>For details about the cause of a slow query, see Viewing Redis Slow Queries.</p>

12.3 Viewing Metrics

The Cloud Eye service monitors the running performance your DCS instances.

Procedure

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** Click the desired instance.
- Step 5** Choose **Performance Monitoring**. All monitoring metrics of the instance are displayed.

NOTE

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

----End

12.4 Configuring Alarm Rules for Critical Metrics

This section describes the alarm rules of some metrics and how to configure the rules. In actual scenarios, configure alarm rules for metrics by referring to the following alarm policies.

Alarm Policies for DCS Redis Instances

Table 12-6 DCS Redis instance metrics to configure alarm rules for

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
CPU Usage	0–100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Major	No	Consider capacity expansion based on the service analysis. The CPU capacity of a single-node or master/standby instance cannot be expanded. If you need larger capacity, use a cluster instance instead. This metric is available only for single-node, master/standby, and Proxy Cluster instances. For Redis Cluster instances, this metric is available only on the Redis Server level. You can view the metric on the Redis Server tab page on the Performance Monitoring page of the instance.
Average CPU Usage	0–100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Major	No	Consider capacity expansion based on the service analysis. The CPU capacity of a single-node or master/standby instance cannot be expanded. If you need larger capacity, use a cluster instance instead. This metric is available only for single-node, master/standby, and Proxy Cluster instances. For Redis Cluster instances, this metric is available only on the Redis Server level. You can view the metric on the Redis Server tab page on the Performance Monitoring page of the instance.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
Memory Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Critical	No	Expand the capacity of the instance.
Connected Clients	0-10,000	Alarm threshold: > 8000 Number of consecutive periods: 2 Alarm severity: Major	No	Optimize the connection pool in the service code to prevent the number of connections from exceeding the maximum limit. Configure this alarm policy on the instance level for single-node and master/standby instances. For cluster instances, configure this alarm policy on the Redis Server and Proxy level. For single-node and master/standby instances, the maximum number of connections allowed is 10,000. You can adjust the threshold based on service requirements.
New Connections (Count/min)	≥ 0	Alarm threshold: > 10,000 Number of consecutive periods: 2 Alarm severity: Minor	-	Check whether connect is used and whether the client connection is abnormal. Use persistent connections (" pconnect " in Redis terminology) to ensure performance. Configure this alarm policy on the instance level for single-node and master/standby instances. For cluster instances, configure this alarm policy on the Redis Server and Proxy level.

Alarm Policies for Redis Server Nodes of Cluster DCS Redis Instances

Table 12-7 Redis server metrics to configure alarm policies for

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
CPU Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Major	No	Check the service for traffic surge. Check whether the CPU usage is evenly distributed to Redis Server nodes. If the CPU usage is high on multiple nodes, consider capacity expansion. Expanding the capacity of a cluster instance will scale out nodes to share the CPU pressure. If the CPU usage is high on a single node, check whether hot keys exist. If yes, optimize the service code to eliminate hot keys.
Average CPU Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Major	No	Consider capacity expansion based on the service analysis. The CPU capacity of a single-node or master/standby instance cannot be expanded. If you need larger capacity, use a cluster instance instead. This metric is available only for single-node, master/standby, and Proxy Cluster instances. For Redis Cluster instances, this metric is available only on the Redis Server level. You can view the metric on the Redis Server tab page on the Performance Monitoring page of the instance.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
Memory Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Major	No	Check the service for traffic surge. Check whether the memory usage is evenly distributed to Redis Server nodes. If the memory usage is high on multiple nodes, consider capacity expansion. If the memory usage is high on a single node, check whether big keys exist. If yes, optimize the service code to eliminate big keys.
Connected Clients	0-10,000	Alarm threshold: > 8000 Number of consecutive periods: 2 Alarm severity: Major	No	Check whether the number of connections is within the appropriate range. If yes, adjust the alarm threshold.
New Connections	≥ 0	Alarm threshold: > 10,000 Number of consecutive periods: 2 Alarm severity: Minor	-	Check whether connect is used. To ensure performance, use persistent connections ("pconnect" in Redis terminology).
Slow Query Logs	0-1	Alarm threshold: > 0 Number of consecutive periods: 1 Alarm severity: Major	-	Use the slow query function on the console to analyze slow commands.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
Bandwidth Usage	0-200%	Alarm threshold: > 90% Number of consecutive periods: 2 Alarm severity: Major	Yes	<p>Check whether the bandwidth usage increase comes from read services or write services based on the input and output flow.</p> <p>If the bandwidth usage of a single node is high, check whether big keys exist.</p> <p>Even if the bandwidth usage exceeds 100%, flow control may not necessarily be performed. The actual flow control is subject to the Flow Control Times metric.</p> <p>Even if the bandwidth usage is below 100%, flow control may be performed. The real-time bandwidth usage is reported once in every reporting period. The flow control times metric is reported every second. During a reporting period, the traffic may surge within seconds and then fall back. By the time the bandwidth usage is reported, it has restored to the normal level.</p>
Flow Control Times	≥ 0	Alarm threshold: > 0 Number of consecutive periods: 1 Alarm severity: Critical	Yes	<p>Consider capacity expansion based on the specification limits, input flow, and output flow.</p>

Alarm Policies for Proxy Nodes of Cluster DCS Redis Instances

Table 12-8 Proxy metrics to configure alarm policies for

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
CPU Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Critical	Yes	Consider capacity expansion, which will add proxies.
Memory Usage	0-100%	Alarm threshold: > 70% Number of consecutive periods: 2 Alarm severity: Critical	Yes	Consider capacity expansion, which will add proxies.
Connected Clients	0-30,000	Alarm threshold: > 20,000 Number of consecutive periods: 2 Alarm severity: Major	No	Optimize the connection pool in the service code to prevent the number of connections from exceeding the maximum limit.

Configuring an Alarm Rule for a Resource Group

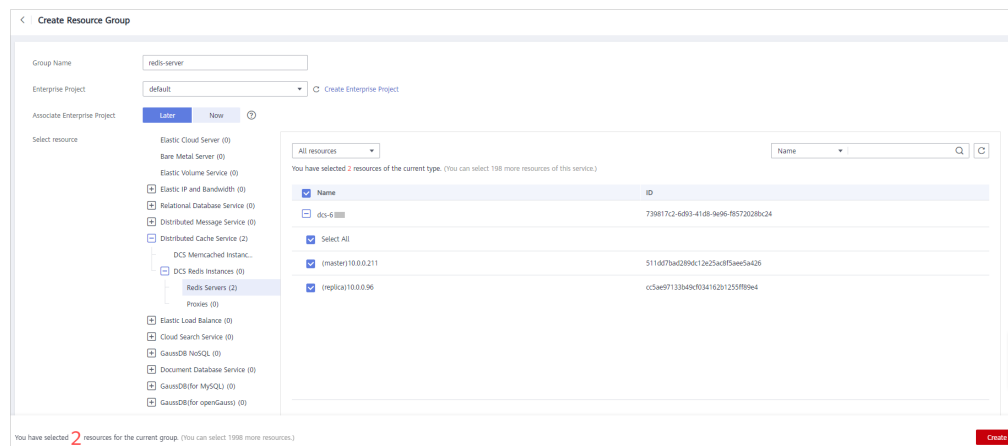
Cloud Eye allows you to add DCS instances, Redis Server nodes, and proxy nodes to resource groups and manage instances and alarm rules by group to simplify O&M. For details, see [Creating a Resource Group](#).

Step 1 Create a resource group.

1. Log in to the Cloud Eye console. In the navigation pane, choose **Resource Groups** and then click **Create Resource Group** in the upper right corner.

2. Enter a group name and add Redis Server nodes to the resource group.
You can add Redis Server nodes of different instances to the same resource group.

Figure 12-1 Creating a resource group

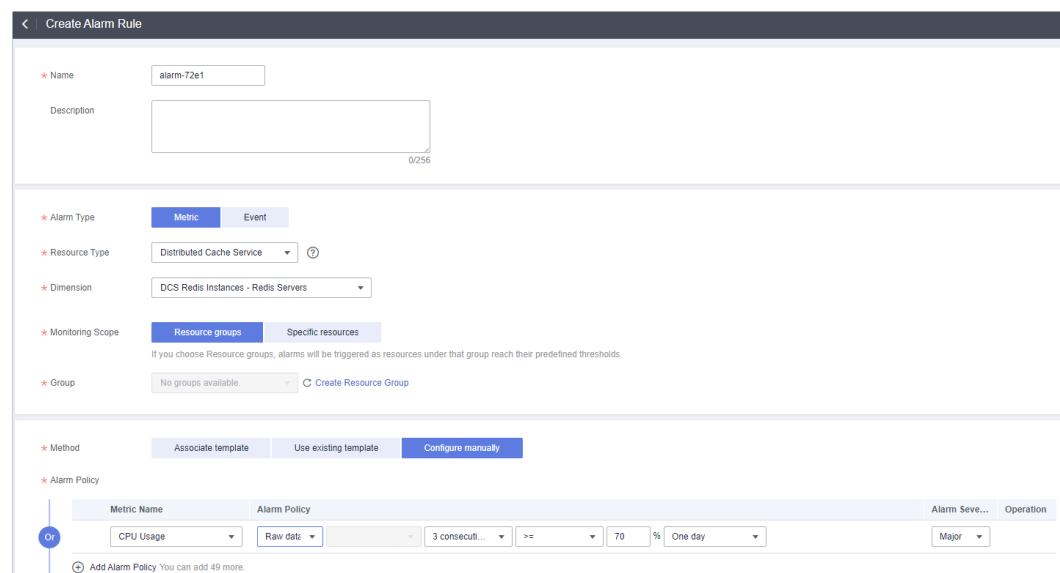


3. Click **Create**.

Step 2 In the navigation pane of the Cloud Eye console, choose **Alarm Management > Alarm Rules** and then click **Create Alarm Rule** to set alarm information for the resource group.

Create a CPU usage alarm rule for all Redis Server nodes in the resource group, as shown in the following figure.

Figure 12-2 Creating an alarm rule for a resource group



Step 3 Click **Create**.

----End

Configuring an Alarm Rule for a Specific Resource

In the following example, an alarm rule is set for the **Slow Query Logs (is_slow_log_exist)** metric.


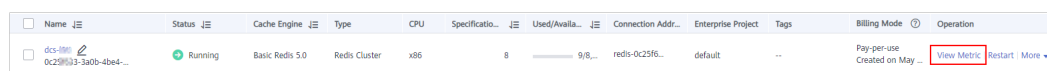

- Step 1** Log in to the management console, and choose **Application > Distributed Cache Service** in the service list.
- Step 2** Click  in the upper left corner of the management console and select the region where your instance is located.
- Step 3** In the navigation pane, choose **Cache Manager**.
- Step 4** In the row containing the DCS instance whose metrics you want to view, click **View Metric** in the **Operation** column.

Figure 12-3 Viewing instance metrics



Name	Status	Cache Engine	Type	CPU	Specifications	Used/Availa...	Connection Addr...	Enterprise Project	Tags	Billing Mode	Operation
dc3-001 0c220333-3a0b-4bed...	Running	Basic Redis 5.0	Redis Cluster	x86	8	9/8...	redis-0c2256...	default	--	Pay-per-use Created on May	View Metric Restart More

- Step 5** On the displayed page, locate the **Slow Query Logs** metric. Hover over the metric and click  to create an alarm rule for the metric.

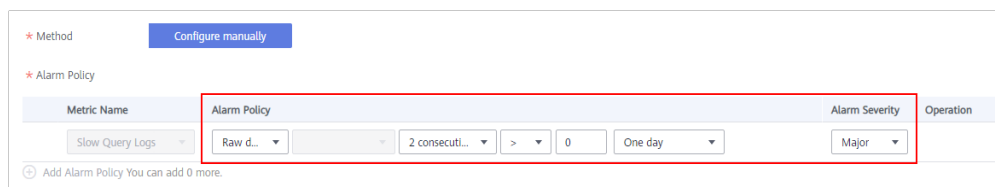
The **Create Alarm Rule** page is displayed.

- Step 6** Specify the alarm information.

1. Set the alarm name and description.
2. Specify the alarm policy and alarm severity.

For example, the alarm policy shown in **Figure 12-4** indicates that an alarm will be triggered if slow queries exist in the instance for two consecutive periods. If no actions are taken, the alarm will be triggered once every day, until the value of this metric returns to **0**.

Figure 12-4 Setting the alarm content



Method: [Configure manually](#)

Alarm Policy

Metric Name	Alarm Policy	Alarm Severity	Operation
Slow Query Logs	Raw d... 2 consecuti... > 0 One day	Major	

⊕ Add Alarm Policy You can add 0 more.

3. Set the alarm notification configurations. If you enable **Alarm Notification**, set the validity period, notification object, and trigger condition.
4. Click **Create**.

NOTE

- For more information about creating alarm rules, see [Creating an Alarm Rule](#).
- To modify or disable alarms, see [Alarm Rule Management](#).

----End

13 Auditing

13.1 Operations Logged by CTS

With CTS, you can query, audit, and review operations performed on cloud resources. Traces include the operation requests sent using the management console or open APIs as well as the results of these requests.

The following lists the DCS operations that can be recorded by CTS.

Table 13-1 DCS operations that can be recorded by CTS

Operation	Resource Type	Trace Name
Creating an instance	Redis	createDCSInstance
Submitting an instance creation request	Redis	submitCreateDCSInstanceRequest
Deleting multiple instances	Redis	batchDeleteDCSInstance
Deleting an instance	Redis	deleteDCSInstance
Modifying instance information	Redis	modifyDCSInstanceInfo
Modifying instance configurations	Redis	modifyDCSInstanceConfig

Operation	Resource Type	Trace Name
Changing instance password	Redis	modifyDCSInstancePassword
Stopping an instance	Redis	stopDCSInstance
Submitting an instance stopping request	Redis	submitStopDCSInstanceRequest
Restarting an instance	Redis	restartDCSInstance
Submitting an instance restarting request	Redis	submitRestartDCSInstanceRequest
Starting an instance	Redis	startDCSInstance
Submitting an instance starting request	Redis	submitStartDCSInstanceRequest
Clearing instance data	Redis	flushDCSInstance
Stopping multiple instances	Redis	batchStopDCSInstance
Submitting a request to stop instances in batches	Redis	submitBatchStopDCSInstanceRequest
Restarting instances in batches	Redis	batchRestartDCSInstance
Submitting a request to restart instances in batches	Redis	submitBatchRestartDCSInstanceRequest

Operation	Resource Type	Trace Name
Starting multiple instances	Redis	batchStartDCSInstance
Submitting a request to start instances in batches	Redis	submitBatchStartDCSInstanceRequest
Restoring instance data	Redis	restoreDCSInstance
Submitting a request to restore instance data	Redis	submitRestoreDCSInstanceRequest
Backing up instance data	Redis	backupDCSInstance
Submitting a request to back up instance data	Redis	submitBackupDCSInstanceRequest
Deleting instance backup files	Redis	deleteInstanceBackupFile
Deleting background tasks	Redis	deleteDCSInstanceJobRecord
Modifying instance specifications	Redis	modifySpecification
Submitting a request to modify instance specifications	Redis	submitModifySpecificationRequest

Operation	Resource Type	Trace Name
Creating an instance subscription order	Redis	createInstanceOrder
Creating an order for modifying instance specifications	Redis	createSpecificationChangeOrder
Updating enterprise project ID	Redis	updateEnterpriseProjectId
Switching between master and standby nodes	Redis	masterStandbySwitchover
Disabling public access	Redis	disablePublicNetworkAccess
Enabling public access	Redis	enablePublicNetworkAccess
Resetting instance password	Redis	resetDCSInstancePassword
Submitting a request to clear instance data	Redis	submitFlushDCSInstanceRequest
Accessing Web CLI	Redis	webCliLogin
Running commands in Web CLI	Redis	webCliCommand
Exiting Web CLI	Redis	webCliLogout
Migrating offline data	Redis	offlineMigrate

Operation	Resource Type	Trace Name
Changing the billing mode	Redis	billingModeChange
Updating instance tags	Redis	updateInstanceTag
Modifying the whitelist configuration	Instance	modifyWhiteList

13.2 Querying Real-Time Traces


Scenarios




After you enable CTS and the management tracker is created, CTS starts recording operations on cloud resources. After a data tracker is created, the system starts recording operations on data in OBS buckets. CTS stores operation records generated in the last seven days.

This section describes how to query and export operation records of the last seven days on the CTS console.


- [Viewing Real-Time Traces in the Trace List of the New Edition](#)
- [Viewing Real-Time Traces in the Trace List of the Old Edition](#)



Viewing Real-Time Traces in the Trace List of the New Edition

1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & Deployment** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. On the **Trace List** page, use advanced search to query traces. You can combine one or more filters.
 - **Trace Name:** Enter a trace name.
 - **Trace ID:** Enter a trace ID.
 - **Resource Name:** Enter a resource name. If the cloud resource involved in the trace does not have a resource name or the corresponding API operation does not involve the resource name parameter, leave this field empty.
 - **Resource ID:** Enter a resource ID. Leave this field empty if the resource has no resource ID or if resource creation failed.
 - **Trace Source:** Select a cloud service name from the drop-down list.

- **Resource Type:** Select a resource type from the drop-down list.
 - **Operator:** Select one or more operators from the drop-down list.
 - **Trace Status:** Select **normal**, **warning**, or **incident**.
 - **normal:** The operation succeeded.
 - **warning:** The operation failed.
 - **incident:** The operation caused a fault that is more serious than the operation failure, for example, causing other faults.
 - Time range: Select **Last 1 hour**, **Last 1 day**, or **Last 1 week**, or specify a custom time range.
5. On the **Trace List** page, you can also export and refresh the trace list, and customize the list display settings.
 - Enter any keyword in the search box and press Enter to filter desired traces.
 - Click **Export** to export all traces in the query result as an .xlsx file. The file can contain up to 5000 records.
 - Click  to view the latest information about traces.
 - Click  to customize the information to be displayed in the trace list. If **Auto wrapping** is enabled (), excess text will move down to the next line; otherwise, the text will be truncated. By default, this function is disabled.
 6. For details about key fields in the trace structure, see section "Trace References" > "Trace Structure" and section "Trace References" > "Example Traces".
 7. (Optional) On the **Trace List** page of the new edition, click **Go to Old Edition** in the upper right corner to switch to the **Trace List** page of the old edition.

Viewing Real-Time Traces in the Trace List of the Old Edition

1. Log in to the management console.
2. Click  in the upper left corner and choose **Management & Deployment** > **Cloud Trace Service**. The CTS console is displayed.
3. Choose **Trace List** in the navigation pane on the left.
4. Each time you log in to the CTS console, the new edition is displayed by default. Click **Go to Old Edition** in the upper right corner to switch to the trace list of the old edition.
5. Set filters to search for your desired traces. The following filters are available:
 - **Trace Type**, **Trace Source**, **Resource Type**, and **Search By:** Select a filter from the drop-down list.
 - If you select **Resource ID** for **Search By**, specify a resource ID.
 - If you select **Trace name** for **Search By**, specify a trace name.
 - If you select **Resource name** for **Search By**, specify a resource name.

- **Operator:** Select a user.
 - **Trace Status:** Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.
 - Time range: You can query traces generated during any time range in the last seven days.
 - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
6. Click **Query**.
 7. On the **Trace List** page, you can also export and refresh the trace list.
 - Click **Export** to export all traces in the query result as a CSV file. The file can contain up to 5000 records.
 - Click  to view the latest information about traces.
 8. Click  on the left of a trace to expand its details.

Trace Name	Resource Type	Trace Source	Resource ID	Resource Name	Trace Status	Operator	Operation Time	Operation
createDockerConfig	dockerlogincmd	SWR	-	dockerlogincmd	normal		Nov 16, 2023 10:54:04 GMT+08:00	View Trace

```

request
trace_id: [redacted]
code: 200
trace_name: createDockerConfig
resource_type: dockerlogincmd
trace_rating: normal
api_version:
message: createDockerConfig, Method: POST Url=/v2/management/secret, Reason:
source_ip: [redacted]
domain_id: [redacted]
trace_type: ApiCall
            
```

9. Click **View Trace** in the **Operation** column. The trace details are displayed.

View Trace ×

```

{
  "request": "",
  "trace_id": "[redacted]",
  "code": "200",
  "trace_name": "createDockerConfig",
  "resource_type": "dockerlogincmd",
  "trace_rating": "normal",
  "api_version": "",
  "message": "createDockerConfig, Method: POST Url=/v2/management/secret, Reason:",
  "source_ip": "[redacted]",
  "domain_id": "[redacted]",
  "trace_type": "ApiCall",
  "service_type": "SWR",
  "event_type": "system",
  "project_id": "[redacted]",
  "response": "",
  "resource_id": "",
  "tracker_name": "system",
  "time": "Nov 16, 2023 10:54:04 GMT+08:00",
  "resource_name": "dockerlogincmd",
  "user": {
    "domain": {
      "name": "[redacted]",
      "id": "[redacted]"
    }
  }
}
            
```

10. For details about key fields in the trace structure, see section "Trace References" > "Trace Structure" and section "Trace References" > "Example Traces" in the *CTS User Guide*.
11. (Optional) On the **Trace List** page of the old edition, click **New Edition** in the upper right corner to switch to the **Trace List** page of the new edition.