Distributed Cache Service

User Guide

Issue	01
Date	2024-12-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions

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 Process of Using DCS

How to Manage DCS Instances

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 console

  You can sign up and log in to the console, click ☰ in the upper left corner on the homepage, and choose **Middleware** > **Distributed Cache Service for Redis**.

  For details on how to use the DCS console, see chapters from **Buying a DCS Redis Instance** to **Migrating Instance Data**.

  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 DCS 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 **Viewing DCS Audit Logs**.

- 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. For DCS instance functions with open APIs, manage them using the console or calling the APIs. For those without open APIs, manage them using the console.
2. For details about APIs for monitoring and auditing, see the **Cloud Eye** and **Cloud Trace Service (CTS)** documentation.

2 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.

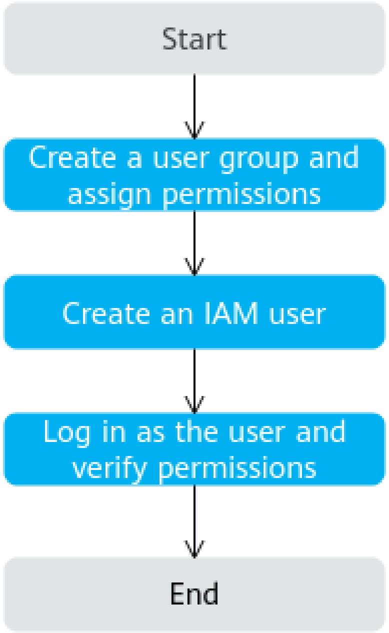
This section describes the procedure for granting the **DCS ReadOnlyAccess** permission (see **Figure 2-1**) as an example.

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


Start
Create a user group and assign permissions
Create an IAM user
Log in as the user and verify permissions
End

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.

DCS Custom Policies

You can create custom policies 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: Create a JSON policy or edit an existing one.

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: Grant permission 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: Grant permission to deny DCS deletion.

  A policy with only "Deny" permissions must be used together with other policies. If the permissions granted to an IAM user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

  Assume that you want to grant the permissions of the **DCS FullAccess** policy to a user but want to prevent them from deleting DCS instances. You can create a custom policy for denying DCS deletion, and attach this policy together with the **DCS FullAccess** policy to the user. As an explicit deny in any policy overrides any allows, the user can perform all operations on DCS instances excepting deleting them. Example policy denying DCS deletion:

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

DCS Resources

A resource is an object that exists within a service. DCS resources include instance. To select these resources, specify their paths.

Table 2-1 DCS resources and their paths

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

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.

Preparing Required Resources

DCS Redis instances are deployed in Virtual Private Clouds (VPCs), and bound to specific subnets. In this way, Redis instances are isolated with virtual networks that users can manage by themselves.

Therefore, before creating a Redis instance, prepare a VPC and subnet if you do not have one yet.

Table 3-1 Dependency resources of a DCS instance

Resource	Requirement	Operations
VPC and subnet	Different Redis instances can use the same or different VPCs and subnets as required. Note the following when creating a VPC and subnet: ● 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**.

Buying a DCS Redis Instance

Step 1 Go to the **Buy DCS Instance** page.

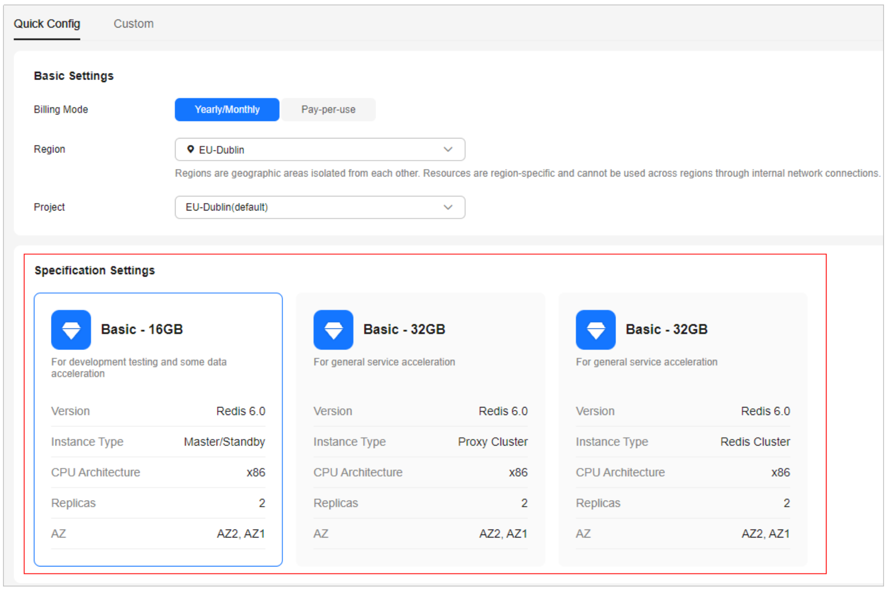
Step 2 Choose a purchase mode. **Quick Config** and **Custom** are available.

In **Quick Config**, several common **Specification Settings** are available, as shown in . If none of them meet your needs, go to the **Custom** tab page to customize instance types and specifications, as shown in .

NOTE

Quick Config and **Custom** differ only in specification selection.

Figure 3-1 Quick Config



Basic Settings

Billing Mode: Yearly/Monthly | Pay-per-use

Region: EU-Dublin

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections.

Project: EU-Dublin(default)

Specification Settings

Basic - 16GB
For development testing and some data acceleration

Version	Redis 6.0
Instance Type	Master/Standby
CPU Architecture	x86
Replicas	2
AZ	AZ2, AZ1

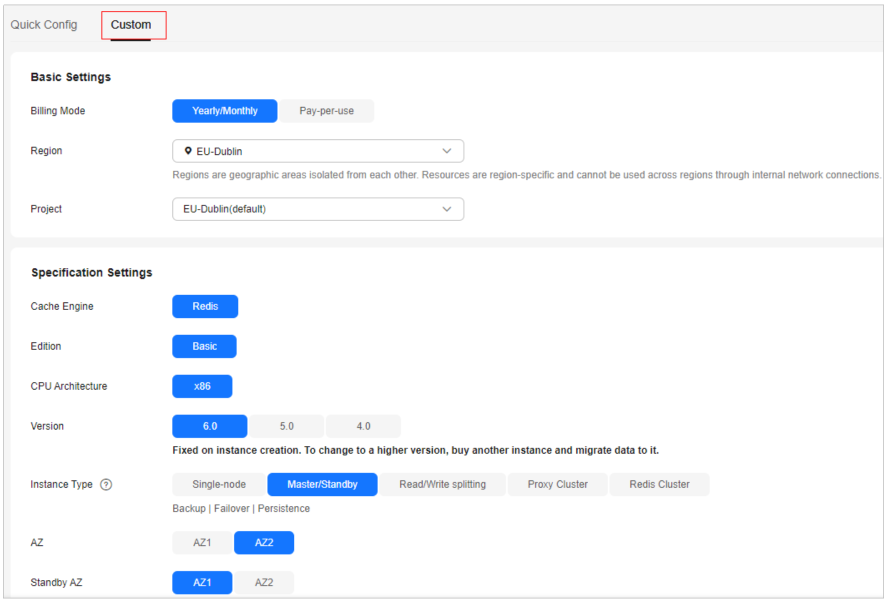
Basic - 32GB
For general service acceleration

Version	Redis 6.0
Instance Type	Proxy Cluster
CPU Architecture	x86
Replicas	2
AZ	AZ2, AZ1

Basic - 32GB
For general service acceleration

Version	Redis 6.0
Instance Type	Redis Cluster
CPU Architecture	x86
Replicas	2
AZ	AZ2, AZ1

Figure 3-2 Custom



Basic Settings

Billing Mode: Yearly/Monthly | Pay-per-use

Region: EU-Dublin

Regions are geographic areas isolated from each other. Resources are region-specific and cannot be used across regions through internal network connections.

Project: EU-Dublin(default)

Specification Settings

Cache Engine: Redis

Edition: Basic

CPU Architecture: x86

Version: 6.0 | 5.0 | 4.0

Fixed on instance creation. To change to a higher version, buy another instance and migrate data to it.

Instance Type: Single-node | Master/Standby | Read/Write splitting | Proxy Cluster | Redis Cluster

Backup | Failover | Persistence

AZ: AZ1 | AZ2

Standby AZ: AZ1 | AZ2

Step 3 Specify **Billing Mode**.

Step 4 Select a region closest to your application to reduce latency and accelerate access.

Step 5 Select a project. By default, each region corresponds to a project.

Step 6 Configure specifications. If you choose **Quick Config**, select one of the common specifications by referring to **Table 3-2**. If you choose **Custom**, configure instance specifications by referring to **Table 3-3**.

Table 3-2 Specifications (Quick Config)

Item	Description
Basic - memory/ Professional - memory	Edition and memory. For example, **Basic - 16GB** is a basic edition instance with 16 GB memory.
Version	Version of the Redis instance. **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.
Instance Type	Master/Standby, Redis Cluster, and Proxy Cluster instances can be quickly configured. For details, see **DCS Instance Types**.
CPU Architecture	x86-based CPU can be quickly configured.
Replicas	Two replicas mean that the instance has two nodes (one master and one standby).
AZ	AZs where the master node and standby node of the instance are located.

Table 3-3 Specifications (Custom)

Item	Description
Cache Engine	Only Redis is available.
CPU Architecture	**x86** is available.
Version	Currently supported Redis versions: 4.0, 5.0, and 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.
Instance Type	Single-node, master/standby, Proxy Cluster, Redis Cluster, and read/write splitting types are supported. For more information, see **DCS Instance Types**.

Item	Description
AZ	**AZ**: If **Instance Type** is master/standby, read/write splitting, Proxy Cluster, or Redis Cluster, **AZ** and **Standby AZ** are displayed. In this case, select AZs for the master and standby nodes of the instance. Each region consists of multiple AZs with physically isolated power supplies and networks. The master and standby nodes of a master/standby, read/write splitting, or cluster DCS instance can be deployed in different AZs. Applications can also be deployed across AZs to achieve high availability (HA) for both data and applications. **NOTE** ● If instance nodes in an AZ are faulty, nodes in other AZs will not be affected. 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. ● To accelerate access, deploy your instance and your application in the same AZ.
Replicas	Enter the number of replicas. Replicas are DCS instance nodes. One replica indicates only a master node. Two replicas indicate a master node and a standby node. Three replicas indicate a master node and two standby nodes. Replica quantity range varies by version and instance type. **Replicas** cannot be set for single-node instances.
Sharding	This parameter is available only for cluster instances, and is single-choice. The shard size and quantity cannot be specified at once. ● **Use default**: You do not need to specify the shard size and quantity. Select one of the default instance specifications. ● **Custom shard size**: Select a shard size. Then select an instance specification. ● **Custom shard quantity**: Select a shard quantity. Then select an instance specification.
Instance Specification	In the **Instance Specification** area, select memory as required. For more information about the instance performance, see **DCS Instance Specifications**. The default memory quota is displayed on the console.

Step 7 Configure instance network settings.

1. Select the created **VPC** and **Subnet**.

NOTE

- To access the instance in an Elastic Cloud Server (ECS), select the VPC where the ECS is.
- The VPC and subnet are fixed once the DCS instance is created.

2. In the **IPv4 Address** area, set the instance (private) IP address.

   Redis Cluster and professional edition 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 an IP address for your instance as required.

3. Configure **Port**. 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.
4. Basic edition Redis instances are based on VPC endpoints and do not support security groups. You can **configure a whitelist** after creating an instance.

Step 8 Set **Instance Name**.

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 **dcs_demo**, the two instances are respectively named as **dcs_demo-000** and **dcs_demo-001**.

Step 9 Specify **Enterprise Project**. An enterprise project manages cloud resources by gathering relevant ones together.

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 Set the instance password.

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

  NOTE

  - Password-free access carries security risks. Exercise caution when selecting this mode.
  - After creating a password-free DCS Redis instance, you can set a password for it later by using the password reset function. 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**.

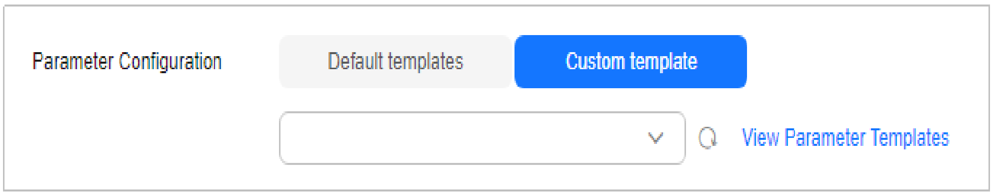
  NOTE

  - 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. The system cannot detect your password.

Step 11 Click **Advanced Settings** and set the following information as required.

1. Configure **Parameter Configuration**. Retain **Default templates** or select **Use custom template** as required.

   If you select **Use custom template**, select one from the drop-down list box. To view or modify the configuration in the selected template, click **View Parameter**. If no custom parameter template of the selected instance version and type is available, the selection box is empty. In this case, click **View Parameter Templates** to go to the template creation page to create a template. For details, see **Creating a Custom Parameter Template for a DCS Instance**.

2. To configure instance backup policies, enable **Auto Backup**.

   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**.

3. Currently, **Public Access** cannot be configured during instance creation.
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**.

5. 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.

6. 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.

7. Enter a description of the instance.

Step 12 Specify **Required Duration**. Determine the purchase duration and whether to enable auto-renewal only when purchasing a yearly/monthly Redis instance.

Step 13 Specify **Quantity**.

Step 14 Click **Next**.

The displayed page shows the instance information you have specified.

Step 15 Confirm the instance information and submit the request.

Step 16 After the task is successfully submitted, the **Cache Manager** page is displayed. When the new instance is in the **Running** state, the instance is created successfully.

----End

4 Accessing a DCS Redis Instance

4.1 Configuring Redis Network Connections

4.1.1 Network Conditions for Accessing DCS Redis

You can access a DCS instance through any Redis client. For details about Redis clients, see the **Redis official website**.

There are different constraints when a client connects to Redis in certain ways:

- Accessing a Redis instance on a client within the same VPC

  The ECS where the 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**.

- Accessing a Redis instance on a client across VPCs in the same region

  If the client 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?**

- Accessing a Redis instance of another region on a client

  If the client server and the Redis instance are not in the same region, connect the network using Direct Connect. For details, see **What Is Direct Connect**.

  To access a Redis instance across regions, the instance domain names cannot be resolved across regions. Therefore, the instance cannot be accessed at its domain name addresses. You can manually map the domain name to the IP address in the **hosts** file, or access the instance at its IP address.

4.2 Controlling DCS Redis Access

4.2.1 Configuring DCS Redis Access Whitelist

The following describes how to manage whitelists of a Redis instance to allow access only from whitelisted IP addresses. Enabling whitelists only allows instance access from IP addresses within them, and only applies to new connections.

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 4-1 Whitelist parameters

Parameter	Description	Example
Group Name	Whitelist group name of the instance. A maximum of four whitelist groups can be created for each instance. Group naming rules: ● Start with a letter. ● 4 to 64 characters. ● Only letters, digits, hyphens (-), and underscores (_) are allowed.	DCS-test
IP Address/Range	IP addresses or address segments of the instances allowed for access. A maximum of 100 IP addresses or IP address segments can be added to an instance. Use commas (,) to separate multiple IP addresses or address segments. Unsupported IP address and IP address range: 0.0.0.0 and 0.0.0.0/0.	10.10.10.1,10.10.10.10,192.168.0.0/16

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

- To modify a whitelist: Click **Edit** on the whitelist page to modify the IP addresses or address segments of a whitelist.
- To delete a whitelist: Click **Delete** on the whitelist page to delete a whitelist group.
- To disable a whitelist: Click **Disable Whitelist** in the left corner of the whitelist tab page. After a whitelist is disabled, IP addresses within the same VPC as the instance can be used to access the instance. To enable the whitelist, click **Enable Whitelist**.

----End

4.2.2 Configuring a Redis Password

For security purposes, DCS provides password-protected instances. In addition, Redis can be accessed without a password. Use an instance access mode as required.

For a DCS instance that is used on the live network or contains important information, you are advised to set a password.

- To modify an instance password, see **Changing an Instance Password**.
- To change the access mode (password-protected or password-free), or to reset the password, see **Resetting an Instance Password**.

Suggestions for Password Security

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 **-a** ***<password>*** when running commands in redis-cli. After connecting to Redis, run the **auth** command to complete authentication. For 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.

Changing an Instance Password

NOTE

- You cannot change the password of a DCS instance in password-free mode.
- The desired Redis instance 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.)

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 The **Change Password** dialog box is displayed. Enter the old and new password, and confirm it.

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

Resetting an Instance Password

NOTE

- The instance must be in the **Running** state.
- Disabling password protection may compromise security. You can set a password later by password resetting.
- For security purposes, password-free access must be disabled when public access is enabled.
- 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 old password of the instance is still being used.
- Resetting passwords takes effect immediately without server 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.)

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:

- Change password-protected access to password-free access.

  Switch the toggle for **Password-Free Access** and click **OK**.

- Change password-free access to password-protected access or reset the password.

  Enter a password, confirm the password, and click **OK**.

  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 (`~!@#$^&*()-_=+\|{},<.>/?)

----End

4.2.3 Transmitting DCS Redis Data with Encryption Using SSL

Single-node, master/standby, and Redis Cluster DCS Redis 6.0 instances support SSL encryption to ensure data transmission security. This function is not available for other instance versions. RESP (Redis Serialization Protocol), the communication protocol of Reids, only supports plaintext transmission in versions earlier than Redis 6.0.

> **NOTE**
>
> Due to SSL encryption, SSL and client IP pass-through cannot be enabled at the same time. Encrypted links do not carry client IPs.

Enabling or Disabling SSL

Step 1 Log in to the DCS console.

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

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

Step 4 On the **Cache Manager** page, click a DCS instance.

Step 5 In the navigation pane, choose **SSL**.

Step 6 Click ✎ next to **SSL Certificate** to enable or disable SSL.

> **NOTICE**
>
> - Enabling or disabling SSL will restart the instance and disconnect it for a few seconds. Wait until off-peak hours and ensure that your application can re-connect.
> - The restart cannot be undone. For single-node DCS instances and other instances where AOF persistence is disabled (**appendonly** is set to **no**), data will be cleared and ongoing backup tasks will be stopped. Exercise caution when performing this operation.
> - Enabling SSL will deteriorate read/write performance.

Step 7 Click **Download Certificate** to download the SSL certificate.

Step 8 Decompress the SSL certificate and upload the decompressed **ca.crt** file to the server where the Redis client is located.

Step 9 Add the path of the **ca.crt** file to the command for connecting to the instance. For example, to access an instance on redis-cli, see **Connecting to Redis on redis-cli**.

----End

4.2.4 Configuring DCS Redis ACL Users

If you need multiple accounts for a Redis instance, use ACL to create users. ACL users support read-only or read/write permissions.

Prerequisites

By default, this function is supported by DCS Redis 4.0/5.0 instances.

Configuring DCS Redis ACL Users

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}***.

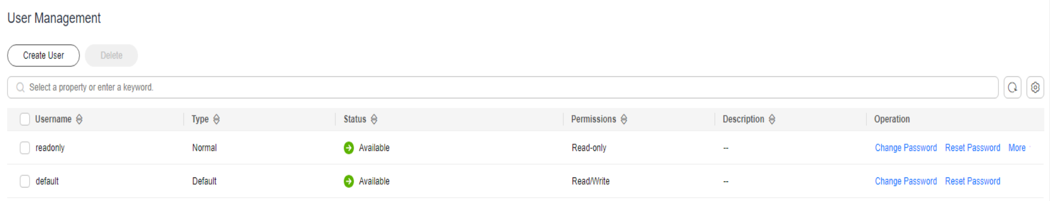
- When **using redis-cli** to connect to an instance, the default user runs the following command:

  ./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}***

Figure 4-1 User management



User Management

Username	Type	Status	Permissions	Description	Operation
readonly	Normal	Available	Read-only	--	Change Password Reset Password More
default	Default	Available	Read/Write	--	Change Password Reset Password

----End

More Operations

The following operations can be performed on normal users.

Table 4-2 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.

4.3 Connecting to Redis on a Client

4.3.1 Connecting to Redis on redis-cli

This section describes how to access a DCS Redis instance on redis-cli. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.

  Run the following command to install GCC on the ECS if needed:

  ```
  yum install -y make
  yum install -y pcre-devel
  yum install -y zlib-devel
  yum install -y libevent-devel
  ```

```
yum install -y openssl-devel
yum install -y gcc-c++
```

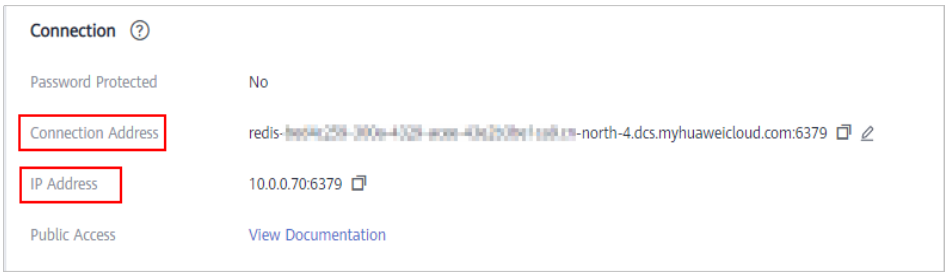
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on redis-cli (Linux)

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

See **Viewing Instance Details** to obtain **Connection Address** or **IP Address** in the instance information. For more information, see .

Figure 4-2 Obtaining the instance addresses



Connection

Password Protected: No

Connection Address: redis-[redacted]-north-4.dcs.myhuaweicloud.com:6379

IP Address: 10.0.0.70:6379

Public Access: View Documentation

NOTE

- The following uses the port 6379. Replace 6379 with the actual port.
- For Proxy Cluster Redis 4.0 and later instances, **Connection Address** and **IP Address** are load balancer addresses. The system distributes requests to different proxies.

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
```

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 {port}
```

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

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

```
[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:

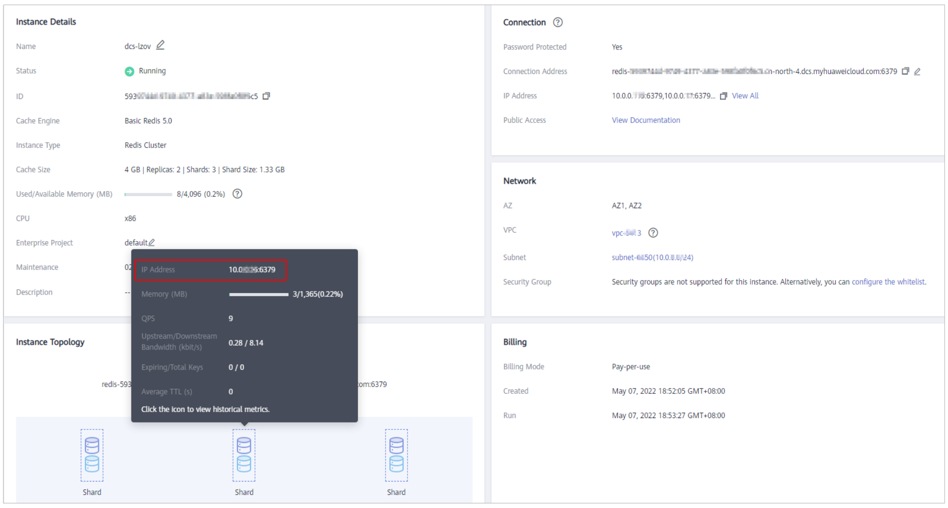
```
./redis-cli -h {dcs_instance_address} -p {port} -a {password} -c
```

{dcs_instance_address} indicates the IP address/domain name of the DCS Redis instance, **{port}** 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

- **When connecting to a Redis Cluster instance using redis-cli, ensure that -c is added to the command.** Otherwise, the connection will fail.
- To connect to Redis on a client over a private network, 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 4-3**.
- For a password-protected instance, you do not need to enter the instance access password **-a {password}**. If you have forgotten the password or need to reset the password, see **Resetting an Instance Password**.
- 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 4-3 Obtaining the addresses for connecting to a Redis Cluster DCS instance



Instance Details

Name dcs-lzov

Status Running

ID 593[redacted]c5

Cache Engine Basic Redis 5.0

Instance Type Redis Cluster

Cache Size 4 GB | Replicas: 2 | Shards: 3 | Shard Size: 1.33 GB

Used/Available Memory (MB) 8/4,096 (0.2%)

CPU x86

Enterprise Project default

Maintenance 02

Description --

IP Address 10.0[redacted]:6379

Memory (MB) 3/1,365(0.22%)

QPS 9

Upstream/Downstream Bandwidth (kbit/s) 0.28 / 8.14

Expiring/Total Keys 0 / 0

Average TTL (s) 0

Click the icon to view historical metrics.

Instance Topology

redis-593 om:6379

Connection

Password Protected Yes

Connection Address redis-[redacted]n-north-4.dcs.myhuaweicloud.com:6379

IP Address 10.0.0.[redacted]:6379,10.0.0.[redacted]:6379... View All

Public Access View Documentation

Network

AZ AZ1, AZ2

VPC vpc-[redacted]3

Subnet subnet-[redacted]50(10.0.[redacted]24)

Security Group Security groups are not supported for this instance. Alternatively, you can configure the whitelist.

Billing

Billing Mode Pay-per-use

Created May 07, 2022 18:52:05 GMT+08:00

Run May 07, 2022 18:53:27 GMT+08:00

- 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 **cluster nodes** command to view the Redis Cluster node information:

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
0988ae8fd3686074c9afdcce73d7878c81a33ddc 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

Connecting to Redis on redis-cli (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**.

4.3.2 Connecting to Redis on Jedis (Java)

This section describes how to access a Redis instance on Jedis. 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.

NOTE

Springboot 2.3.12.RELEASE or later is required. Jedis **3.10.0** or later is required.

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed. For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

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>${jedis.version}<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 4-3 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	-	Redis instance password

Table 4-4 RedisClusterConfiguration parameters

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

Table 4-5 JedisPoolConfig parameters

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

Parameter	Default Value	Description
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.
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 4-6 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) × 150% = 300

4.3.3 Connecting to Redis on Lettuce (Java)

This section describes how to access a Redis instance on Lettuce. 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.

NOTE

Springboot 2.3.12.RELEASE or later is required. Lettuce **6.3.0.RELEASE** or later is required.

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed. For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

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>
<dependency>
    <groupId>io.lettuce</groupId>
    <artifactId>lettuce-core</artifactId>
    <version>${lettuce.version}</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
```

- 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

- 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.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))
                .readFrom(ReadFrom.MASTER)
                .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 clientConfiguration = 
LettucePoolingClientConfiguration.builder()
                .poolConfig(poolConfig())
                .commandTimeout(Duration.ofMillis(redisReadTimeout))
                .clientOptions(clientOptions)
                .readFrom(ReadFrom.MASTER)
                .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))
                .readFrom(ReadFrom.MASTER)
                .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)
                .readFrom(ReadFrom.MASTER)
                .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 4-7 LettuceConnectionFactory parameters

Parameter	Type	Default Value	Description
configuration	RedisConfiguration	-	Redis connection configuration. Two subsclasses: ● 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 4-8 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 4-9 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 4-10 LettuceClientConfiguration parameters

Parameter	Type	Default Value	Description
timeout	Duration	60s	Command timeout: Recommended: **2s**.
clientOptions	ClientOptions	-	Configuration options.
readFrom	readFrom	MASTER	Read mode. Recommended: **MASTER**. Other values may cause access failures in failover scenarios.

Table 4-11 LettucePoolingClientConfiguration parameters

Parameter	Type	Default Value	Description
timeout	Duration	60s	Command timeout: Recommended: **2s**.
clientOptions	ClientOptions	-	Configuration options.
poolConfig	GenericObjectPoolConfig	-	Connection pool configuration.
readFrom	readFrom	MASTER	Read mode. Recommended: **MASTER**. Other values may cause access failures in failover scenarios.

Table 4-12 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**.
cancelCommandsOnReconnectFailure	boolean	true	Indicates whether to cancel commands after a failed reconnection attempt. Recommended: **false**.

Parameter	Type	Default Value	Description
disconnectedBehavior	DisconnectedBehavior	DisconnectedBehavior.DEFAULT	Indicates what to do when a connection drops. Recommended: **ACCEPT_COMMANDS**. ● **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 4-13 SocketOptions parameters

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

Table 4-14 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.
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**.

Parameter	Default Value	Description
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 pockets 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 OSs' kernels, the service system remains unavailable for a long time. **Therefore, you are advised to use Jedis instead of Lettuce.**

4.3.4 Connecting to Redis on Redisson (Java)

This section describes how to access a Redis instance on Redisson. 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.
- Springboot 2.3.12.RELEASE or later is required. Redisson 3.37.0 or later is required.

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed. For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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);
        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(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);
    }
}
```

Parameter Description

Table 4-15 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 4-16 SingleServerConfig parameters (single-node, read/write splitting,, or Proxy Cluster)

Parameter	Default Value	Description
address	-	Node connection information, in redis://*ip*:*port* 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.
subcriptionPerConnection	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 4-17 MasterSlaveServersConfig parameters (master/standby)

Parameter	Default Value	Description
masterAddress	-	Master node connection information, in redis://*ip*:*port* format.
slaveAddresses	-	Standby node connection information, in Set<redis://*ip*:*port*> 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**. Other values may cause access failures in failover scenarios.
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.
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.
subcriptionPerConnection	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**.

Parameter	Default Value	Description
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 4-18 ClusterServersConfig parameters (Redis Cluster)

Parameter	Default Value	Description
nodeAddress	-	Connection addresses of cluster nodes. Each address uses the redis://*ip*:*port* 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**. Other values may cause access failures in failover scenarios.
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.
subcriptionPerConnection	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**

  Similar to **readMode**, **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) × 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.

4.3.5 Connecting to Redis on redis-py (Python)

This section describes how to access a Redis instance on redis-py. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

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 Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.

- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on redis-py

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

For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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 ECS OS 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.

- **For single-node, master/standby, or Proxy Cluster types:**
  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:

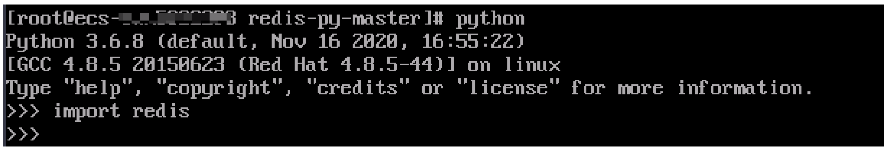
     ```
     wget https://github.com/andymccurdy/redis-py/archive/master.zip
     unzip master.zip
     ```

     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 4-4 Running the python command

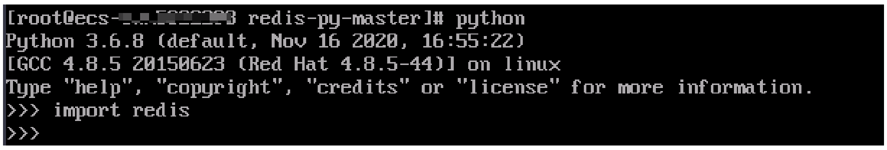


```
[root@ecs-[redacted] redis-py-master]# python
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

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 4-5 Entering the CLI mode

```
[root@ecs-[redacted] 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
>>>
```

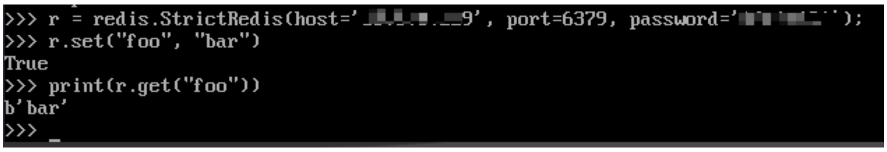
   ii. Run the following command to access the chosen DCS Redis instance:

```
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 4-6 Redis connected successfully

```
>>> r = redis.StrictRedis(host='[redacted]9', port=6379, password='[redacted]');
>>> r.set("foo", "bar")
True
>>> print(r.get("foo"))
b'bar'
>>> _
```

- **For the Redis Cluster type:**

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

4.3.6 Connecting to Redis on go-redis (Go)

This section describes how to access a Redis instance on go-redis. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- View the IP address/domain name and port number of the DCS Redis instance to be accessed. For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.
- An ECS has been created. For details about how to create an ECS, see **Purchasing an ECS**
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on go-redis

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

4.3.7 Connecting to Redis on hiredis (C++)

This section describes how to access a Redis instance on hiredis (C++). For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

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 Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on hiredis

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

For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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

4.3.8 Connecting to Redis on StackExchange.Redis (C#)

This section describes how to access a Redis instance on StackExchange.Redis. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

NOTE

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

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on StackExchange.Redis

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

For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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.

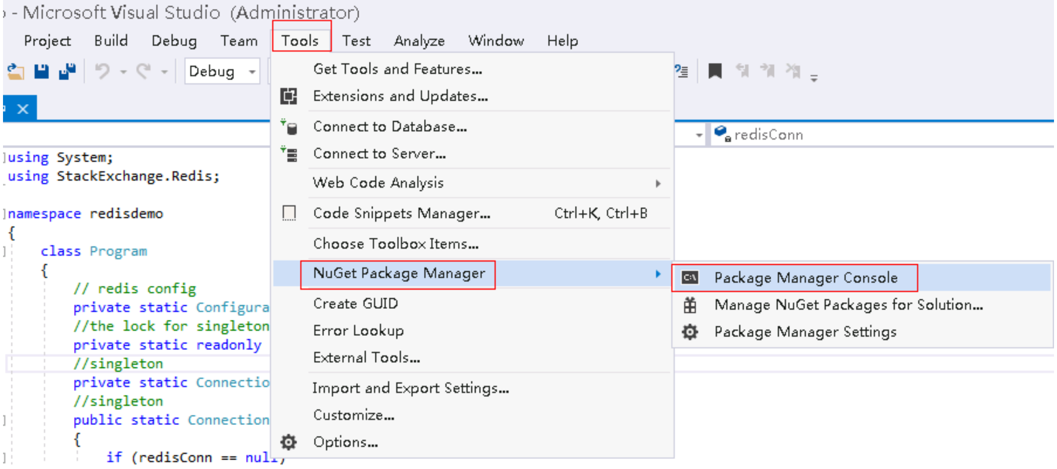
Step 4 Start Visual Studio 2017 and create a project.

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 4-7**, and enter **Install-Package StackExchange.Redis -*Version 2.2.79*.** (The version number is optional).

Figure 4-7 Accessing the NuGet package manager console



```
using System;
using StackExchange.Redis;

namespace redisdemo
{
    class Program
    {
        // redis config
        private static Configura
        //the lock for singleton
        private static readonly
        //singleton
        private static Connectio
        //singleton
        public static Connection
        {
            if (redisConn == nul
```

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

4.3.9 Connecting to Redis on phpredis (PHP)

This section describes how to connect to Redis on phpredis. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

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 Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on phpredis

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

For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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

4.3.10 Connecting to Redis on predis (PHP)

This section describes how to connect to Redis on predis. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on predis

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

For details, see **Viewing and Modifying Basic Settings of a DCS Instance**.

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

4.3.11 Connecting to Redis on ioredis (Node.js)

This section describes how to access a Redis instance on ioredis. For more information about how to use other Redis clients, visit **the Redis official website**.

The following operations are based on an example of accessing a Redis instance on a client on an elastic cloud server (ECS).

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To access a Redis Cluster instance on ioredis, see **Node.js Redis client description**.

Prerequisites

- A Redis instance is created, and is in the **Running** state. To create a Redis instance, see **Buying a DCS Redis Instance**.
- 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.
- The client and the Redis instance must be interconnected before connecting to the instance. For details, see **Network Conditions for Accessing DCS Redis**.

Connecting to Redis on ioredis

- **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 and Modifying Basic Settings of a DCS Instance**.

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 Redis 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 Redis 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 and Modifying Basic Settings of a DCS Instance**.

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 Redis 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 Redis 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.

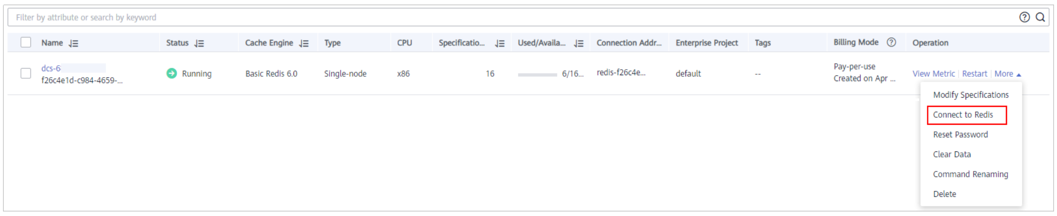
Connecting to Redis on the Console

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**. In the **Operation** column of the instance, choose **More** > **Connect to Redis**, as shown in the following figure.

Figure 4-8 Accessing Web CLI



Filter by attribute or search by keyword

☐	Name	Status	Cache Engine	Type	CPU	Specificatio...	Used/Availa...	Connection Addr...	Enterprise Project	Tags	Billing Mode	Operation
☐	dcs-6[redacted] f26c4e1d-c984-4659-...	Running	Basic Redis 6.0	Single-node	x86	16	6/16...	redis-f26c4e...	default	--	Pay-per-use Created on Apr ...	View Metric | Restart | More

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 Managing Instances

5.1 Viewing and Modifying Basic Settings of a DCS Instance

On the DCS console, you can view and modify DCS instance basic information.

Viewing and Modifying Basic Information of a DCS Instance

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 version of DCS. For example, Redis 4.0. The cache version is fixed once the instance is created. To use another version, create an instance again and migrate the data.
	Instance Type	Type of the selected instance. Currently, supported types include single-node, master/standby, Proxy Cluster, read/write splitting, and Redis Cluster. To change the instance type, see **Modifying DCS Instance Specifications** about the supported instance types, changing notes and procedure.
	Cache Size	Specification of the chosen instance. To change the instance specification, see **Modifying DCS Instance Specifications** about the changing notes and procedure.
	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: ● 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. Select a new window from the drop-down list and click ✓ to save, or ✗ to cancel. The modification takes effect immediately.

Section	Parameter	Description
	Description	Description of the chosen DCS instance. To modify the description, click the ✎ icon.
Connection	Password Protected	**Yes**: password-protected access; **No**: password-free access. To change the password access mode, see **Configuring a Redis Password**.
	Connection Address	**The domain name and port of the Redis instance to be accessed on a client within the VPC.** You can click ✎ next to **Connection Address** to change the port. The connection address is fixed once the instance is created. **NOTE** ● 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	**The IP address and port of the DCS instance to be accessed on a client within the VPC.** To change the instance port, click ✎. The IP address is fixed once the instance is created. The domain name address is recommended.
	Public Access	Currently, public access cannot be enabled for Redis instances.
Network	AZ	Availability zone in which the instance nodes running the selected DCS instance reside.
	VPC	VPC in which the chosen instance resides. The VPC is fixed once the instance is created.
	Subnet	Subnet in which the chosen instance resides. The subnet is fixed once the instance is created.
	Security Group	To control access to DCS Redis instances, **configure a whitelist**
Instance Topology	-	Hover over a node to view its metrics, or click the icon of a node to view its historical metrics. Single-node instances do not display the instance topology.

Section	Parameter	Description
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 Viewing DCS 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.

Viewing DCS Background Tasks

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 Choose **Background Tasks**.

Filter tasks by specifying the time, property, or keyword.

- 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

5.3 Viewing Client Connection Information of a DCS Instance

You can view the client connection information of a DCS instance and disconnect clients.

Notes and Constraints

The session management page displays only the information about the external client connections. Information about the Web CLI connections is not displayed.

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 to go to the details page.

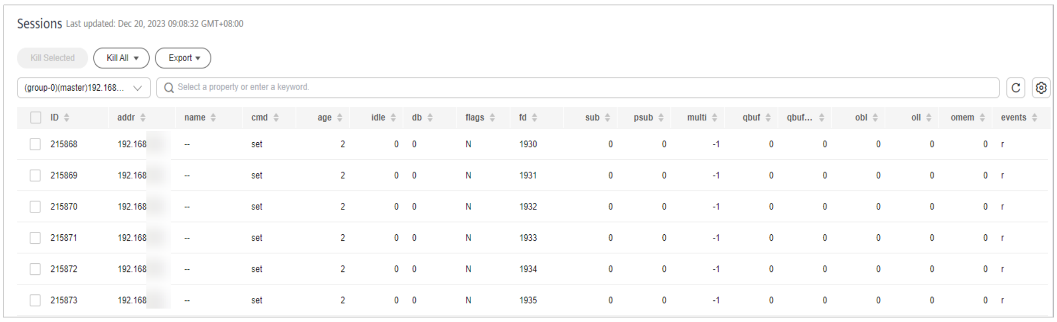
Step 5 Click the **Sessions** tab.

Step 6 Information about client connections of the instance is displayed.

NOTE

- For Proxy Cluster and read/write splitting instances, connections to proxy nodes are displayed. For single-node, master/standby, and Redis Cluster instances, connections to Redis Server nodes are displayed.
- On the page, you can specify a Redis Server or proxy node to query, enter an address, update the query results, and set columns to display.

Figure 5-1 Managing sessions


Kill Selected
Kill All ▾
Export ▾
(group-0)(master)192.168...
Select a property or enter a keyword.

Sessions Last updated: Dec 20, 2023 09:08:32 GMT+08:00

ID	addr	name	cmd	age	idle	db	flags	fd	sub	psub	multi	qbuf	qbuf...	obl	oll	omem	events
□ 215868	192.168[redacted]	--	set	2	0	0	N	1930	0	0	-1	0	0	0	0	0	r
□ 215869	192.168[redacted]	--	set	2	0	0	N	1931	0	0	-1	0	0	0	0	0	r
□ 215870	192.168[redacted]	--	set	2	0	0	N	1932	0	0	-1	0	0	0	0	0	r
□ 215871	192.168[redacted]	--	set	2	0	0	N	1933	0	0	-1	0	0	0	0	0	r
□ 215872	192.168[redacted]	--	set	2	0	0	N	1934	0	0	-1	0	0	0	0	0	r
□ 215873	192.168[redacted]	--	set	2	0	0	N	1935	0	0	-1	0	0	0	0	0	r

Table 5-2 Session fields

Field	Description
ID	Unique ID of a session.

Field	Description
addr	Session address. If IP pass-through is enabled, this address is referred to the client IP address. If not, this address is a private IP address.
name	Client name, which can be configured using **setClientName (...)** in the code. This parameter can be left blank.
cmd	The last command executed.
age	Connection duration, in seconds.
idle	Idle connection duration, in seconds.
db	The DB identifier in the last executed command, for example, the value of **DB0** is **0**.
flags	Connection flags. **M** indicates a connection from a master node. **S** indicates a connection from a standby node. For other flags, see **https://redis.io/docs/latest/commands/client-list/**.
fd	File descriptor.
sub	Number of channel subscriptions.
psub	Number of pattern matching subscriptions.
multi	Number of commands run in transactions or Lua scripts. The value **-1** indicates that no such command is executed.
qbuf	Query buffer length (bytes).
qbuf-free	Free space of the query buffer (bytes).
obl	Output buffer length.
oll	Output list length.
omem	Output buffer memory usage (bytes).
events	File descriptor events (readable, writable). Read: r; Write: w.

Step 7 Select connections to kill and click **Kill Selected** to disconnect the corresponding clients. You can also click **Kill All**.

If a disconnected client can reconnect, it will be automatically reconnected after being disconnected.

Step 8 To export sessions data, click **Export**. You can export all or selected data.

----End

5.4 Modifying Configuration Parameters of a DCS Instance

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

For example, to disable data persistence, set **appendonly** to **no**. For more instance parameters, see **DCS Instance Configuration Parameters**.

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.

Modifying Configuration Parameters of an Instance

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.

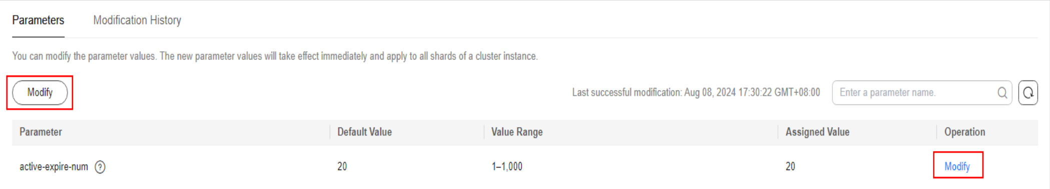
Step 3 In the navigation pane, choose **Cache Manager**.

Step 4 On the **Cache Manager** page, click 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** in the row containing the desired parameter. To modify multiple parameters at a time, click **Modify** above the parameter list.

Figure 5-2 Modifying parameter(s)



You can modify the parameter values. The new parameter values will take effect immediately and apply to all shards of a cluster instance.

Last successful modification: Aug 08, 2024 17:30:22 GMT+08:00

Parameter	Default Value	Value Range	Assigned Value	Operation
active-expire-num ⓘ	20	1–1,000	20	Modify

Step 7 Modify parameters as required.

The parameters are described in **DCS Instance Configuration Parameters**. In most cases, you can retain default values.

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

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

When the parameter modification task is in the **Successful** state, the parameter is modified.

----End

DCS Instance Configuration Parameters

NOTE

- For more information about the parameters described in **Table 5-3**, visit **https://redis.io/topics/memory-optimization**.
- Configurable parameters and their values vary depending on the instance type. If a parameter is not displayed in the **Parameters** page on the console, it cannot be modified.

Table 5-3 DCS Redis instance configuration parameters

Parameter	Description	Exception Scenario	Value Range	Default Value
active-expire-num	Number of randomly checked keys in regular expired key deletions. Enlarging this parameter may increase CPU usage or command latency in a short period of time. Lessening this parameter may increase expired keys in the memory.	-	1–1000	20

Parameter	Description	Exception Scenario	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. The value **0** indicates that the parameter is disabled. That is, the client is not disconnected when it is idle.	-	0–7200 Unit: second	0

Parameter	Description	Exception Scenario	Value Range	Default Value
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.	● 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.	no

Parameter	Description	Exception Scenario	Value Range	Default Value
appendonly	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.	● **yes**: Logs are enabled, that is, persistence is enabled. ● **no**: Logs are disabled, that is, persistence is disabled. ● **only-replica**: Enable persistence only on replica nodes.	yes
client-output-buffer-limit-slave-soft-seconds	When the **client-output-buffer-slave-soft-limit** parameter is exceeded for more than the value of this parameter, the server drops the connection. The smaller the value, the easier the disconnection.	-	0–60 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
client-output-buffer-slave-hard-limit	Hard limit on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit 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. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918

Parameter	Description	Exception Scenario	Value Range	Default Value
maxmemory-policy	The policy applied when the maxmemory limit is reached. 8 values are available.	-	● **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	volatile-lru

Parameter	Description	Exception Scenario	Value Range	Default Value
			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. For details about eviction policies, see the **Redis official website**.	
lua-time-limit	Maximum time allowed for executing a Lua script.	-	100–5,000 Unit: millisecond	5,000
master-read-only	Sets the instance to be read-only. All write operations will fail.	Proxy Cluster instances do not have this parameter.	● yes ● no	no

Parameter	Description	Exception Scenario	Value Range	Default Value
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. The larger the value, the more costly the connection to the server, which affects the server performance and increases the command latency. An excessively small value may constrain the server performance. This parameter specifies the maximum number of connections on a single node (single shard). ● Cluster: Maximum connections per node = Maximum connections of the instance/ Shard quantity ● Single-node and	Read/Write splitting instances do not support this parameter.	1000–50,000	10,000

Parameter	Description	Exception Scenario	Value Range	Default Value
	master/standby: Maximum connections on a single node = Maximum connections of the instance			
proto-max-bulk-len	Maximum size of a single element request. Set this parameter to be greater than the customer request length. Otherwise, the request cannot be executed.	-	1,048,576–536,870,912 Unit: byte	536,870,912

Parameter	Description	Exception Scenario	Value Range	Default Value
repl-backlog-size	The replication backlog size. 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 Unit: byte	1,048,576
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 Unit: second	3,600
repl-timeout	Replication timeout.	Single-node instances do not have this parameter.	30–3,600 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
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

Parameter	Description	Exception Scenario	Value Range	Default Value
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	Exception Scenario	Value Range	Default Value
latency-monitor-threshold	The minimum amount of latency that will be logged as latency spikes If this parameter is set to **0**, latency monitoring is disabled. If this parameter is set to a value greater than 0, all events blocking the server for a time greater than the configured value will be logged. To obtain statistics data, and configure and enable latency monitoring, run the **LATENCY** command.	Proxy Cluster instances do not have this parameter.	0–86,400,000 Unit: millisecond	0

Parameter	Description	Exception Scenario	Value Range	Default Value
	NOTE 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.			

Parameter	Description	Exception Scenario	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. This parameter is disabled only when it is left blank.	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)	Ex

Parameter	Description	Exception Scenario	Value Range	Default Value
			e: Evicted events (events generated when a key is evicted from maxmemory) A: an alias for "g$lshzxe" The parameter value must contain either **K** or **E**. **A** cannot be used together with any of the characters in "g$lshzxe". For example, the value **Kl** 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.	

Parameter	Description	Exception Scenario	Value Range	Default Value
slowlog-log-slower-than	Slow queries cover scheduled commands whose execution is delayed. **slowlog-log-slower-than** is the maximum time allowed for command execution. If this threshold is exceeded, Redis will record the query.	-	0–1,000,000 Unit: microsecond	10,000
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

Parameter	Description	Exception Scenario	Value Range	Default Value
auto-kill-timeout-lua-process	When this parameter is enabled, processes running the lua script 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.	Single-node instances and DCS Redis 3.0 instances do not have this parameter.	● **yes**: enabled ● **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
dispatch-pubsub-to-fixed-shard	This parameter specifies whether pub/sub channels are on the shard of slot 0. When this parameter is enabled, the pub/sub processing logic is consistent with that of single-node instances. You are advised to enable this parameter if you do not depend heavily on pub/sub. If you depend heavily on pub/sub, use the default configuration to allocate subscriptions to all shards.	Only Proxy Cluster instances have this parameter.	● **yes**: Enable this parameter to allocate subscription channels to the shard of slot 0. ● **no**: Disable this parameter to allocate channels to the shard of each channel-hashed slot.	no
readonly-lua-route-to-slave-enabled	If enabled, read-only Lua scripts of read-only users are executed and routed to the standby node.	Only read/ write splitting instances support this parameter.	● **yes**: enabled ● **no**: disabled	no
cluster-sentinel-enabled	To support Sentinels for the instance.	Only Proxy Cluster instances have this parameter.	● **yes**: enabled ● **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
scan-support-wr-split	The **SCAN** command is executed on the master node when this parameter is disabled, or is executed on the standby node otherwise. Enabling this parameter relieves SCAN commands on the master node. But newly written data in the master node may not be synchronized to replicas in time.	Only Proxy Cluster instances have this parameter. Proxy Cluster instances created earlier may not support this parameter. In this case, contact customer service to upgrade instances.	● **yes**: enabled ● **no**: disabled	no

5.5 Configuring DCS Instance Parameter Templates

5.5.1 Viewing a Parameter Template of a DCS Instance

System default parameter templates vary by Redis version and instance type. A system default parameter template contains default instance parameter configurations. Parameter templates can be customized for parameter configurations, and can be selected in instance creation.

This section describes how to view instance 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 management 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 5-4**.

Table 5-4 DCS Redis instance configuration parameters

Parameter	Description	Exception Scenario	Value Range	Default Value
active-expire-num	Number of randomly checked keys in regular expired key deletions. Enlarging this parameter may increase CPU usage or command latency in a short period of time. Lessening this parameter may increase expired keys in the memory.	-	1–1000	20

Parameter	Description	Exception Scenario	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. The value **0** indicates that the parameter is disabled. That is, the client is not disconnected when it is idle.	-	0–7200 Unit: second	0

Parameter	Description	Exception Scenario	Value Range	Default Value
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.	- 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.	no

Parameter	Description	Exception Scenario	Value Range	Default Value
appendonly	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.	- **yes**: Logs are enabled, that is, persistence is enabled. - **no**: Logs are disabled, that is, persistence is disabled. - **only-replica**: Enable persistence only on replica nodes.	yes
client-output-buffer-limit-slave-soft-seconds	When the **client-output-buffer-slave-soft-limit** parameter is exceeded for more than the value of this parameter, the server drops the connection. The smaller the value, the easier the disconnection.	-	0–60 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
client-output-buffer-slave-hard-limit	Hard limit on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit 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. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918

Parameter	Description	Exception Scenario	Value Range	Default Value
maxmemory-policy	The policy applied when the maxmemory limit is reached. 8 values are available.	-	● **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	volatile-lru

Parameter	Description	Exception Scenario	Value Range	Default Value
			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. For details about eviction policies, see the **Redis official website**.	
lua-time-limit	Maximum time allowed for executing a Lua script.	-	100–5,000 Unit: millisecond	5,000
master-read-only	Sets the instance to be read-only. All write operations will fail.	Proxy Cluster instances do not have this parameter.	● yes ● no	no

Parameter	Description	Exception Scenario	Value Range	Default Value
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. The larger the value, the more costly the connection to the server, which affects the server performance and increases the command latency. An excessively small value may constrain the server performance. This parameter specifies the maximum number of connections on a single node (single shard). ● Cluster: Maximum connections per node = Maximum connections of the instance/ Shard quantity ● Single-node and	Read/Write splitting instances do not support this parameter.	1000–50,000	10,000

Parameter	Description	Exception Scenario	Value Range	Default Value
	master/standby: Maximum connections on a single node = Maximum connections of the instance			
proto-max-bulk-len	Maximum size of a single element request. Set this parameter to be greater than the customer request length. Otherwise, the request cannot be executed.	-	1,048,576–536,870,912 Unit: byte	536,870,912

Parameter	Description	Exception Scenario	Value Range	Default Value
repl-backlog-size	The replication backlog size. 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 Unit: byte	1,048,576
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 Unit: second	3,600
repl-timeout	Replication timeout.	Single-node instances do not have this parameter.	30–3,600 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
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

Parameter	Description	Exception Scenario	Value Range	Default Value
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	Exception Scenario	Value Range	Default Value
latency-monitor-threshold	The minimum amount of latency that will be logged as latency spikes If this parameter is set to **0**, latency monitoring is disabled. If this parameter is set to a value greater than 0, all events blocking the server for a time greater than the configured value will be logged. To obtain statistics data, and configure and enable latency monitoring, run the **LATENCY** command.	Proxy Cluster instances do not have this parameter.	0–86,400,000 Unit: millisecond	0

Parameter	Description	Exception Scenario	Value Range	Default Value
	NOTE 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.			

Parameter	Description	Exception Scenario	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. This parameter is disabled only when it is left blank.	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)	Ex

Parameter	Description	Exception Scenario	Value Range	Default Value
			e: Evicted events (events generated when a key is evicted from maxmemory) A: an alias for "g$lshzxe" The parameter value must contain either **K** or **E**. **A** cannot be used together with any of the characters in "g$lshzxe". For example, the value **Kl** 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.	

Parameter	Description	Exception Scenario	Value Range	Default Value
slowlog-log-slower-than	Slow queries cover scheduled commands whose execution is delayed. **slowlog-log-slower-than** is the maximum time allowed for command execution. If this threshold is exceeded, Redis will record the query.	-	0–1,000,000 Unit: microsecond	10,000
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

Parameter	Description	Exception Scenario	Value Range	Default Value
auto-kill-timeout-lua-process	When this parameter is enabled, processes running the lua script 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.	Single-node instances and DCS Redis 3.0 instances do not have this parameter.	● **yes**: enabled ● **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
dispatch-pubsub-to-fixed-shard	This parameter specifies whether pub/sub channels are on the shard of slot 0. When this parameter is enabled, the pub/sub processing logic is consistent with that of single-node instances. You are advised to enable this parameter if you do not depend heavily on pub/sub. If you depend heavily on pub/sub, use the default configuration to allocate subscriptions to all shards.	Only Proxy Cluster instances have this parameter.	- **yes**: Enable this parameter to allocate subscription channels to the shard of slot 0. - **no**: Disable this parameter to allocate channels to the shard of each channel-hashed slot.	no
readonly-lua-route-to-slave-enabled	If enabled, read-only Lua scripts of read-only users are executed and routed to the standby node.	Only read/ write splitting instances support this parameter.	- **yes**: enabled - **no**: disabled	no
cluster-sentinel-enabled	To support Sentinels for the instance.	Only Proxy Cluster instances have this parameter.	- **yes**: enabled - **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
scan-support-wr-split	The **SCAN** command is executed on the master node when this parameter is disabled, or is executed on the standby node otherwise. Enabling this parameter relieves SCAN commands on the master node. But newly written data in the master node may not be synchronized to replicas in time.	Only Proxy Cluster instances have this parameter. Proxy Cluster instances created earlier may not support this parameter. In this case, contact customer service to upgrade instances.	● **yes**: enabled ● **no**: disabled	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 5-4**, visit **https://redis.io/topics/memory-optimization**.

----End

5.5.2 Creating a Custom Parameter Template for a DCS Instance

System default parameter templates vary by Redis version and instance type. A system default parameter template contains default instance parameter configurations. Parameter templates can be customized for parameter configurations, and can be selected in instance creation.

This section describes how to create and modify a custom parameter template on the DCS console.

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 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 5-5 describes the parameters. In most cases, default values are retained.

Table 5-5 DCS Redis instance configuration parameters

Parameter	Description	Exception Scenario	Value Range	Default Value
active-expire-num	Number of randomly checked keys in regular expired key deletions. Enlarging this parameter may increase CPU usage or command latency in a short period of time. Lessening this parameter may increase expired keys in the memory.	-	1–1000	20
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. The value **0** indicates that the parameter is disabled. That is, the client is not disconnected when it is idle.	-	0–7200 Unit: second	0

Parameter	Description	Exception Scenario	Value Range	Default Value
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.	● 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.	no

Parameter	Description	Exception Scenario	Value Range	Default Value
appendonly	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.	● **yes**: Logs are enabled, that is, persistence is enabled. ● **no**: Logs are disabled, that is, persistence is disabled. ● **only-replica**: Enable persistence only on replica nodes.	yes
client-output-buffer-limit-slave-soft-seconds	When the **client-output-buffer-slave-soft-limit** parameter is exceeded for more than the value of this parameter, the server drops the connection. The smaller the value, the easier the disconnection.	-	0–60 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
client-output-buffer-slave-hard-limit	Hard limit on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918
client-output-buffer-slave-soft-limit	Soft limit 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. The smaller the value, the easier the disconnection.	-	0–17,179,869,184 Unit: byte	1,717,986,918

Parameter	Description	Exception Scenario	Value Range	Default Value
maxmemory-policy	The policy applied when the maxmemory limit is reached. 8 values are available.	-	● **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	volatile-lru

Parameter	Description	Exception Scenario	Value Range	Default Value
			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. For details about eviction policies, see the **Redis official website**.	
lua-time-limit	Maximum time allowed for executing a Lua script.	-	100–5,000 Unit: millisecond	5,000
master-read-only	Sets the instance to be read-only. All write operations will fail.	Proxy Cluster instances do not have this parameter.	- yes - no	no

Parameter	Description	Exception Scenario	Value Range	Default Value
maxclients	The maximum number of clients allowed to be concurrently connected to a DCS instance. The larger the value, the more costly the connection to the server, which affects the server performance and increases the command latency. An excessively small value may constrain the server performance. This parameter specifies the maximum number of connections on a single node (single shard). ● Cluster: Maximum connections per node = Maximum connections of the instance/ Shard quantity ● Single-node and	Read/Write splitting instances do not support this parameter.	1000–50,000	10,000

Parameter	Description	Exception Scenario	Value Range	Default Value
	master/standby: Maximum connections on a single node = Maximum connections of the instance			
proto-max-bulk-len	Maximum size of a single element request. Set this parameter to be greater than the customer request length. Otherwise, the request cannot be executed.	-	1,048,576–536,870,912 Unit: byte	536,870,912

Parameter	Description	Exception Scenario	Value Range	Default Value
repl-backlog-size	The replication backlog size. 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 Unit: byte	1,048,576
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 Unit: second	3,600
repl-timeout	Replication timeout.	Single-node instances do not have this parameter.	30–3,600 Unit: second	60

Parameter	Description	Exception Scenario	Value Range	Default Value
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

Parameter	Description	Exception Scenario	Value Range	Default Value
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	Exception Scenario	Value Range	Default Value
latency-monitor-threshold	The minimum amount of latency that will be logged as latency spikes If this parameter is set to **0**, latency monitoring is disabled. If this parameter is set to a value greater than 0, all events blocking the server for a time greater than the configured value will be logged. To obtain statistics data, and configure and enable latency monitoring, run the **LATENCY** command.	Proxy Cluster instances do not have this parameter.	0–86,400,000 Unit: millisecond	0

Parameter	Description	Exception Scenario	Value Range	Default Value
	NOTE 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.			

Parameter	Description	Exception Scenario	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. This parameter is disabled only when it is left blank.	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)	Ex

Parameter	Description	Exception Scenario	Value Range	Default Value
			e: Evicted events (events generated when a key is evicted from maxmemory) A: an alias for "g$lshzxe" The parameter value must contain either **K** or **E**. **A** cannot be used together with any of the characters in "g$lshzxe". For example, the value **Kl** 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.	

Parameter	Description	Exception Scenario	Value Range	Default Value
slowlog-log-slower-than	Slow queries cover scheduled commands whose execution is delayed. **slowlog-log-slower-than** is the maximum time allowed for command execution. If this threshold is exceeded, Redis will record the query.	-	0–1,000,000 Unit: microsecond	10,000
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

Parameter	Description	Exception Scenario	Value Range	Default Value
auto-kill-timeout-lua-process	When this parameter is enabled, processes running the lua script 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.	Single-node instances and DCS Redis 3.0 instances do not have this parameter.	● **yes**: enabled ● **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
dispatch-pubsub-to-fixed-shard	This parameter specifies whether pub/sub channels are on the shard of slot 0. When this parameter is enabled, the pub/sub processing logic is consistent with that of single-node instances. You are advised to enable this parameter if you do not depend heavily on pub/sub. If you depend heavily on pub/sub, use the default configuration to allocate subscriptions to all shards.	Only Proxy Cluster instances have this parameter.	● **yes**: Enable this parameter to allocate subscription channels to the shard of slot 0. ● **no**: Disable this parameter to allocate channels to the shard of each channel-hashed slot.	no
readonly-lua-route-to-slave-enabled	If enabled, read-only Lua scripts of read-only users are executed and routed to the standby node.	Only read/ write splitting instances support this parameter.	● **yes**: enabled ● **no**: disabled	no
cluster-sentinel-enabled	To support Sentinels for the instance.	Only Proxy Cluster instances have this parameter.	● **yes**: enabled ● **no**: disabled	no

Parameter	Description	Exception Scenario	Value Range	Default Value
scan-support-wr-split	The **SCAN** command is executed on the master node when this parameter is disabled, or is executed on the standby node otherwise. Enabling this parameter relieves SCAN commands on the master node. But newly written data in the master node may not be synchronized to replicas in time.	Only Proxy Cluster instances have this parameter. Proxy Cluster instances created earlier may not support this parameter. In this case, contact customer service to upgrade instances.	● **yes**: enabled ● **no**: disabled	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 5-4**, visit **https://redis.io/topics/memory-optimization**.

Step 8 Click **OK**.

----End

Modifying or Deleting Custom Templates

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 To edit a custom parameter template, use either of the following ways:

- Locate the row containing the desired template and click **Edit** in the **Operation** column.
  1. Change the name or modify the description.
  2. In the **Parameters** area, select **Modifiable parameters**. In the row containing the desired parameter, enter a value in the **Assigned Value** column. **Table 5-5** describes the parameters. In most cases, retain the default values.
  3. Click **OK**.
- Click the name of a custom template.
  1. Select **Modifiable parameters**. Enter a keyword in the search box to search for a parameter by its name.
  2. Click **Modify**.
  3. In the row containing the desired parameter, enter a value in the **Assigned Value** column. **Table 5-5** describes the parameters. In most cases, retain the default values.
  4. Click **Save**.

Step 6 To delete custom templates, click **Delete** in the **Operation** column on the right of the templates to be deleted.

Click **Yes**.

----End

5.6 Configuring DCS Instance Tags

Tags facilitate DCS instance identification and management. Tags can be added in instance creation, or on the instance details page. A maximum of 20 tags are allowed for a DCS instance.

A tag consists of a tag key and tag value. **Table 5-6** describes the naming rules for them.

Table 5-6 Tag key and value requirements

Parameter	Requirements
Tag key	• 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 _ . : = + - @ • Cannot start or end with a space. • Cannot start with **_sys_**.

Parameter	Requirements
Tag value	● Consists of a maximum of 255 characters. ● Can contain letters of any language, digits, spaces, and special characters _ . : / = + - @ ● Cannot start or end with a space.

Configuring Instance Tags

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**
  a. 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**.

  b. 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

5.7 Renaming Critical Commands for DCS Instances

Certain high-risk commands can be modified for DCS Redis instances. Once a command is modified, it is only known to the modifier. The original command is blocked to be run by other users.

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.

NOTE

- Only DCS Redis 4.0 and later instances support command renaming.
- An instance will restart when its commands are renamed. Exercise caution.
- Renaming takes effect immediately once it is complete. Renamed commands will not be displayed on the console for security purposes.
- A command can be renamed multiple times. Each new name overwrites the previous name. To restore a high-risk command, or if a renamed command is forgotten, rename the command.
- A command cannot be renamed to other original commands. For example, **KEYS** can be renamed to **KEYS** or **ABC123**, but cannot be renamed to **SCAN**.
- Renaming a command starts only with a letter and contains 4–64 characters of letters, digits, hyphens (-), and underscores (_).

Renaming Critical Commands

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.

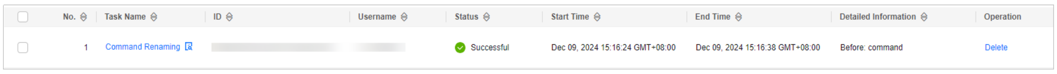
Step 3 In the navigation pane, choose **Cache Manager**.

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	ID	Username	Status	Start Time	End Time	Detailed Information	Operation
□	1	Command Renaming	[redacted]	[redacted]	✓ Successful	Dec 09, 2024 15:16:24 GMT+08:00	Dec 09, 2024 15:16:38 GMT+08:00	Before: command	Delete

----End

5.8 Exporting a DCS Instance List

A list of instance information can be exported and downloaded in Excel on the DCS console. You can view or compare DCS instance information.

Procedure

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

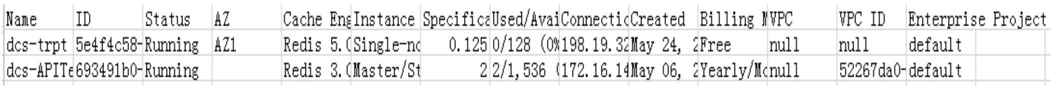
Step 2 Click in the upper left corner of the management console and select the region where your instance is located.

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

Step 4 Click **Export** to export all instances by default. To export some instances, select them and click **Export**.

Step 5 The **Tasks** page is displayed. When the **Export DCS instance list** task is in the **Successful** state, click **Download** on the right to download the list.

Figure 5-4 Exported DCS instance list



Name	ID	Status	AZ	Cache Eng	Instance	Specifica	Used/Avai	Connectic	Created	Billing M	VPC	VPC ID	Enterprise Project
dcs-trpt	5e4f4c58-	Running	AZ1	Redis 5.(	Single-nc	0.125	0/128 (0%	198.19.32	May 24, 2	Free	null	null	default
dcs-APITe	693491b0-	Running		Redis 3.(	Master/St	2	2/1,536 (	172.16.14	May 06, 2	Yearly/Mc	null	52267da0-	default

----End

5.9 Performing a Master/Standby Switchover for a DCS Instance

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 DCS Instance Shards and Replicas**.

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.

Performing a Master/Standby Switchover for a DCS Instance

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**.

Step 5 When the master/standby switchover task is in the **Successful** state, the master/standby is switched over.

----End

5.10 Managing DCS Instance Shards and Replicas

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 for a DCS Instance**.
- On the **Node Management** page: The failover priority can be edited for master/standby instances with multiple replicas. The IP address of a replica can be removed (only when multiple replicas exist). The information returned when an instance is accessed at read-only domain names excludes the removed IP addresses.
- 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

- 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 adjust shards of a cluster instance by referring to **Modifying DCS Instance Specifications**.

Managing DCS Instance Shards and Replicas

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**.

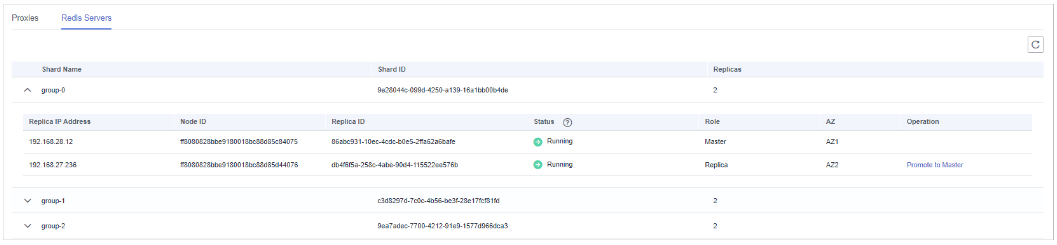
Step 4 Click an instance.

Step 5 Click **Node Management**.

All shards of the instance are displayed by **Shard Name**, **Shard ID**, and **Replicas** of a shard.

Step 6 Click ⌄ to display all replicas of a shard by **Replica IP Address**, **Node ID**, **Replica ID**, **Status**, **Role**, and **AZ**.

Figure 5-5 Node management (cluster instance)

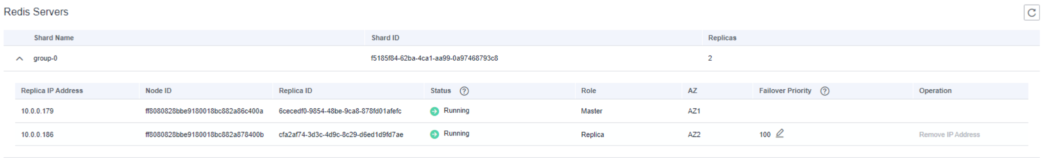


Shard Name	Shard ID	Replicas
group-0	9e28044c-099d-4250-a139-16a1bb00b4de	2

Replica IP Address	Node ID	Replica ID	Status	Role	AZ	Operation
192.168.28.12	ff8080828bbe9180018bc88d85c84075	86abc931-10ec-4cdc-b0e5-2ffa62a6bafe	Running	Master	AZ1	
192.168.27.236	ff8080828bbe9180018bc88d85d44076	db4f6f5a-258c-4abe-90d4-115522ee576b	Running	Replica	AZ2	Promote to Master

Shard Name	Shard ID	Replicas
group-1	c3d8297d-7c0c-4b56-be3f-28e17fcf81fd	2
group-2	9ea7adec-7700-4212-91e9-1577d966dca3	2

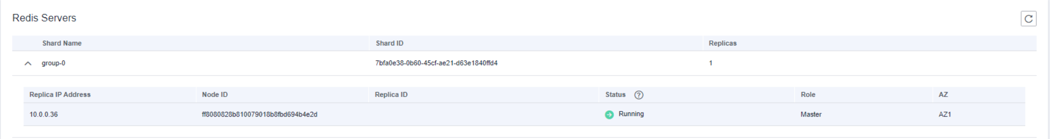
Figure 5-6 Node management (master/standby instance)



Shard Name	Shard ID	Replicas
group-0	f5185f84-62ba-4ca1-aa99-0a97468793c8	2

Replica IP Address	Node ID	Replica ID	Status	Role	AZ	Failover Priority	Operation
10.0.0.179	ff8080828bbe9180018bc882a86c400a	6cecedf0-9854-48be-9ca8-878fd01afefc	Running	Master	AZ1		
10.0.0.186	ff8080828bbe9180018bc882a878400b	cfa2af74-3d3c-4d9c-8c29-d6ed1d9fd7ae	Running	Replica	AZ2	100 ✎	Remove IP Address

Figure 5-7 Node management (single-node instance)



Shard Name	Shard ID	Replicas
group-0	7bfa0e38-0b60-45cf-ae21-d63e1840ffd4	1

Replica IP Address	Node ID	Replica ID	Status	Role	AZ
10.0.0.36	ff8080828b810079018b8fbd694b4e2d		Running	Master	AZ1

You can also perform the following operations on replicas:

- 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

  For **Proxy Cluster** instance, the proxy information (IP address, node ID, and name) can be viewed on the **Node Management** > **Proxies** page. Other types of instances do not have the **Proxies** tab page.

- Master/Standby or read/write splitting

  a. 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.

  b. 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 Backing Up or Restoring Instance Data

6.1 DCS Backup and Restoration 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.

Impact on DCS Instances During Backup

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

In the event of full synchronization of master and standby nodes 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.

New data changes on the master node during an ongoing backup are not included in the backup.

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.
- 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 the most recent backup record 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.
- Exercise caution when deleting all backup files.

Restoring Data

- 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.

6.2 Backing up DCS Instances Automatically

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.

Configuring Automated Backup Policies

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 instance 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 6-1 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**.

The automated backup can be disabled. The backup policies can be modified.

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

6.3 Backing up DCS Instances Manually

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.
- Exercise caution when deleting all backup files.

Prerequisites

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

Backing up DCS Instances Manually

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 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

6.4 Restoring DCS Instances

This section describes how to restore instances on the DCS console. This function helps restore instances deleted by mistake.

To migrate backup data to other DCS instances, see **Backup Import Between DCS Redis Instances**.

Notes and Constraints

- 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.
- This function becomes unavailable after the following changes.
  - Instance scale-in
  - Cluster instance scale-out
  - Instance type changes (from master/standby to read/write splitting not included)

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 instance 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

6.5 Downloading DCS Instance Backup Files

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.

Downloading DCS Instance Backup Files

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**.

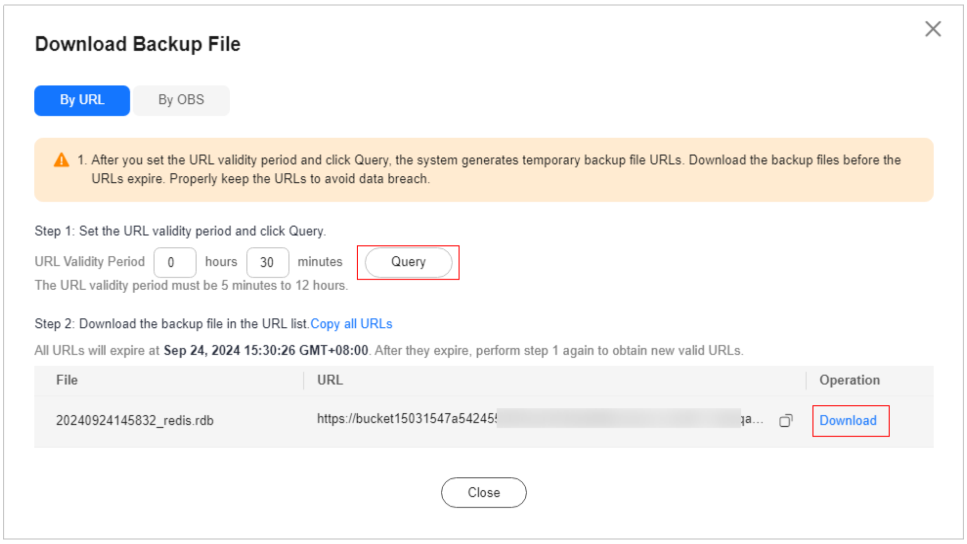
A list of historical backup tasks is then displayed.

Step 6 Click **Download** in the row containing the chosen backup task.

Step 7 Download the backup by URL or OBS.

- By URL

Figure 6-1 By URL



Download Backup File

By URL | By OBS

1. After you set the URL validity period and click Query, the system generates temporary backup file URLs. Download the backup files before the URLs expire. Properly keep the URLs to avoid data breach.

Step 1: Set the URL validity period and click Query.

URL Validity Period 0 hours 30 minutes Query

The URL validity period must be 5 minutes to 12 hours.

Step 2: Download the backup file in the URL list. Copy all URLs

All URLs will expire at **Sep 24, 2024 15:30:26 GMT+08:00**. After they expire, perform step 1 again to obtain new valid URLs.

File	URL	Operation
20240924145832_redis.rdb	https://bucket15031547a54245[redacted]ja...	Download

Close

a. Set the URL validity period and click **Query**. A temporary backup file URL will be generated. The URL has a validity period, and needs to be generated again after that.

URL validity period: 5 minutes to 12 hours.

b. In the URL list, click **Download** to download files.

NOTE

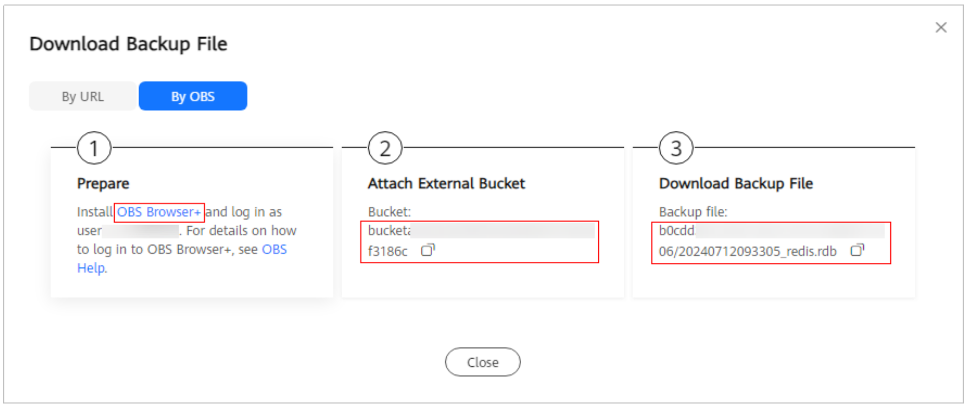
- Click **Copy all URLs** or the copy icon after a URL to copy URLs.
- If you choose to copy URLs, use quotation marks to quote the URLs when running the **wget** command in Linux. For example:

  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.
- Keep the backup files and URLs secure to prevent data leakage.

- By OBS

Figure 6-2 By OBS



Download Backup File

By URL | By OBS

1 Prepare

Install OBS Browser+ and log in as user [redacted]. For details on how to log in to OBS Browser+, see OBS Help.

2 Attach External Bucket

Bucket:
bucket[redacted]
f3186c

3 Download Backup File

Backup file:
b0cdd[redacted]
06/20240712093305_redis.rdb

Close

a. Click **By OBS**, click **OBS Browser+** in the **Prepare** area.

b. Install the OBS Browser+ client. Log in to OBS Browser+ using the Huawei Cloud account by referring to **Logging In to OBS Browser+**.

c. Add an external bucket to the OBS Browser+ client. For details, see **Adding an External Bucket**.

   Use the bucket name in the **Attach External Bucket** area.

d. Click the bucket name. Search for backup files in the bucket. For details, see **Searching for a File or Folder**.

   The backup file path in the **Download Backup File** area contains the name of the backup file and folder.

e. Click the download icon on the right to download the backup file.

----End

7 Changing an Instance

7.1 Modifying DCS Instance Specifications

On the DCS console, you can change DCS Redis instance specifications including the instance type, memory, and replica quantity.

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 customer service 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 7-1 Instance type change options supported by different DCS instances

Version	Supported Type Change	Precautions
Redis 4.0/5.0/6.0	From master/standby or read/write splitting to Proxy Cluster	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, create an instance, migrate data, and switch IPs. For details, see **Migrating Instance Data**.

For details about the commands supported by different types of instances, see **Command Compatibility**.

Scaling

- **Scaling options**

Table 7-2 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, 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, out/in	Scaling up/down and replica quantity change
Redis 6.0	Scaling up/down	Scaling up/down and replica quantity change	-	-	-

- **Impact of scaling**

Table 7-3 Impact of scaling

Instance Type	Scaling Type	Impact
Single-node , read/write splitting, and master/standby	Scaling up/down	- During scaling up, a DCS Redis 4.0 or later 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	● Scaling out by adding shards: – **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: – 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 increasing the size per shard: – **Insufficient memory of the node's VM will cause the node to migrate. Service connections may stutter and the instance may become read-only during the migration.** – Increasing the node capacity when the VM memory is sufficient does not affect services. ● 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**

Instance Type	Scaling Type	Impact
		enabled. If it is disabled, you will need to restart the client after scaling. For details about how to enable automated refresh if you use Lettuce, see **an example of using Lettuce to connect to a Redis Cluster instance**. ● Backup records created before scaling cannot be used. If necessary, download the backup file in advance or back up the data again after scaling.
Master/ Standby, read/ write splitting, and Redis Cluster instances	Scaling out/in (replica quantity change)	● 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.

Changing an Instance

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** of the **Operation** column in the row containing the DCS instance.

Step 5 On the **Modify Specifications** page, select the desired specification.

Step 6 Click **Next**. Confirm the change details and view the risk check results.

If any risk is found in the check, the instance may fail to be modified. For details, see **Table 7-4**.

Table 7-4 Risk check items

Check Item	Reason for Check	Solution
Non-standard configuration check **NOTE** - Check whether the following items meet standards: – Bandwidth of a single instance node – Memory of a single instance node – Replica quantity of Redis Cluster instances – Proxy quantity of Proxy Cluster instances – **maxclients** of Proxy Cluster instances (maximum allowed connections exceeded)	If your instance has non-standard configurations, the console displays a message indicating that they will be converted to standard during the change. You can retain non-standard bandwidth or proxy quantity configuration only.	- If your instance does not have non-standard configurations, the check result is normal and no action is required. - If the instance has non-standard configurations, determine whether to proceed with the change or whether to retain the non-standard bandwidth and proxy quantity configuration.
Node status	Abnormal instance nodes cause instance modification failures.	If this case, contact customer service.
Dataset memory distribution check **NOTE** This check item applies only to Proxy Cluster and Redis Cluster instances.	Specification modification of a cluster instance involves data migration between nodes. If an instance has any key bigger than 512 MB, the modification will fail when big key migration between nodes times out. If the instance dataset memory is unevenly distributed among nodes and the difference is greater than 512 MB, the instance has a big key and the change may fail.	before proceeding with the change.
Memory usage check	If the memory usage of a node is greater than 90%, keys may be evicted or the change may fail.	If the memory usage is too high, optimize the memory by optimizing big keys, scanning for expired keys, or deleting some keys.

Check Item	Reason for Check	Solution
Network input traffic check **NOTE** This check item applies only to single-node, read/write splitting, and master/standby instances.	The change may fail if the network input traffic is too heavy and the write buffer overflows.	Perform the change during off-peak hours.
CPU usage check	If the node CPU usage within 5 minutes is greater than 90%, the change may fail.	Perform the change during off-peak hours.
Resource capacity **NOTE** This item should be checked only when scaling up cluster instances.	To scale up a cluster instance, if the VM resource capacity is insufficient, the node needs to be migrated during the change. Service connections become intermittent or read-only during the migration.	If the resource capacity check poses risks, ensure that your application can reconnect to Redis or handle exceptions, you may need to restart the application after the change.

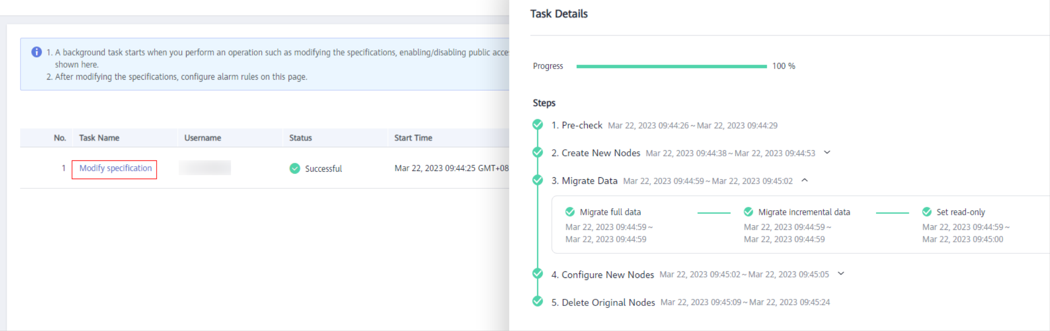
NOTE

- If the check results are normal, no risks are found in the check.
- If the check fails, the possible causes are as follows:
  - The master node of the instance fails to be connected. In this case, check the instance status.
  - The system is abnormal. In this case, click **Check Again** later.
- Click **Stop Check** to stop the check. Click **Check Again** to restart the check.
- If you want to proceed with the change despite risks found in the check or after clicking **Stop Check**, select **I understand the risks.**

Step 7 Click **Next**, confirm the details, and click **Submit**. After the modification is submitted, you can go to the **Background Tasks** page to view the modification status.

Click the task name on the **Background Tasks** page to view task details. After an instance is successfully modified, it changes to the **Running** state.

Figure 7-1 Viewing background task details



1. A background task starts when you perform an operation such as modifying the specifications, enabling/disabling public acce
shown here.
2. After modifying the specifications, configure alarm rules on this page.

No.	Task Name	Username	Status	Start Time
1	Modify specification	[redacted]	Successful	Mar 22, 2023 09:44:25 GMT+08

Task Details

Progress 100 %

Steps

1. Pre-check Mar 22, 2023 09:44:26 ~ Mar 22, 2023 09:44:29
2. Create New Nodes Mar 22, 2023 09:44:38 ~ Mar 22, 2023 09:44:53
3. Migrate Data Mar 22, 2023 09:44:59 ~ Mar 22, 2023 09:45:02
   - Migrate full data Mar 22, 2023 09:44:59 ~ Mar 22, 2023 09:44:59
   - Migrate incremental data Mar 22, 2023 09:44:59 ~ Mar 22, 2023 09:44:59
   - Set read-only Mar 22, 2023 09:44:59 ~ Mar 22, 2023 09:45:00
4. Configure New Nodes Mar 22, 2023 09:45:02 ~ Mar 22, 2023 09:45:05
5. Delete Original Nodes Mar 22, 2023 09:45:09 ~ Mar 22, 2023 09:45:24

NOTE

- If the specification modification of a single-node DCS instance fails, the instance is temporarily unavailable. 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 still uses 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.
- Specification modification of a single-node, master/standby, or read/write splitting DCS instance takes 5 to 30 minutes to complete, while that of a cluster DCS instance takes a longer time.

----End

7.2 Adjusting DCS Instance Bandwidth

Generally, Redis instances save and obtain data in the data layer closer to application services, which consumes the network bandwidth. Rate limits may occur when the instance bandwidth is insufficient, causing increased service latency or client connection exceptions. Currently, the Redis instance bandwidth can be adjusted on the console for DCS Redis 4.0 and later instances.

Notes and Constraints

- This function is unavailable for professional edition DCS Redis instances.
- This function is available only for instances in the **Running** state.
- The bandwidth adjustment range is from the instance's assured bandwidth to its maximum bandwidth. Generally, the maximum bandwidth per shard is 2048 Mbit/s when the physical machine of the instance node has sufficient bandwidths.
- The relationship between the instance bandwidth and the bandwidth of a single shard is as follows:
  - Bandwidth of single-node or master/standby instances = Bandwidth per shard

- – Bandwidth of read/write splitting instances = Bandwidth per shard × Replica quantity
- – Bandwidth of cluster instances = Bandwidth per shard × Shard quantity, or the total bandwidth of all shards if the bandwidth per shard varies

Procedure

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

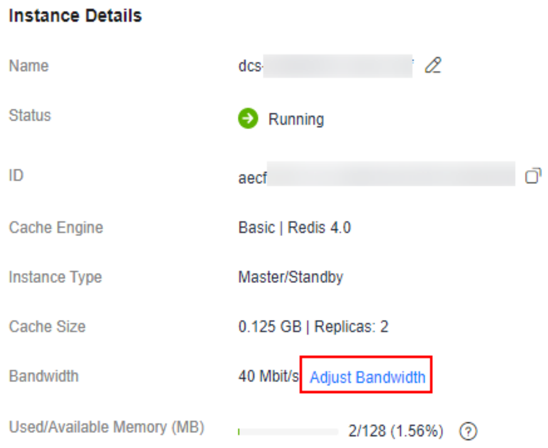
Step 2 Click in the upper left corner of the management console and select the region where your instance is located.

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

Step 4 On the **Cache Manager** page, click the name of the DCS instance you want to configure.

Step 5 In the **Instance Details** area of the DCS instance, click **Adjust Bandwidth** next to **Bandwidth**.

Figure 7-2 Adjusting bandwidth



Instance Details

Name	dcs[redacted]
Status	Running
ID	aecf[redacted]
Cache Engine	Basic | Redis 4.0
Instance Type	Master/Standby
Cache Size	0.125 GB | Replicas: 2
Bandwidth	40 Mbit/s Adjust Bandwidth
Used/Available Memory (MB)	2/128 (1.56%)

Step 6 The **Adjust Bandwidth** page is displayed. Specify bandwidth.

For cluster instances with multiple shards, adjust the bandwidth for each shard or select the desired shards and click **Adjust Bandwidth**.

NOTE

- Set the bandwidth to a multiple of 8 within the valid range. Otherwise, a value rounded down to a multiple of 8 will be automatically used after the order is submitted. For example, if you set the bandwidth to 801, 800 will be used instead.
- **Price** shown on the change page is only the fee of the additional bandwidth.
- The additional bandwidth is pay-per-use.
- You can adjust the bandwidth whenever as required. If you perform multiple bandwidth changes in a billing period (one hour), you will be billed according to highest bandwidth in the period. For example, if you have changed the bandwidth of a DCS Redis instance from 256 Mbit/s (default) to 2048 Mbit/s, and changed the bandwidth again to 512 Mbit/s in the same billing period, you will be billed at the price of 2048 Mbit/s bandwidth.

Step 7 Confirm the bandwidth and fees, check **Authorization**, and click **Submit**.

When the bandwidth adjustment task is in the **Successful** state, the new bandwidth is used.

----End

7.3 Changing Cluster DCS Instances to be Across AZs

To implement disaster recovery, cluster instances (whose master and standby nodes are) in a single AZ can be deployed across AZs by migrating the standby nodes to other AZs.

Notes and Constraints

- Available only for single-AZ cluster instances with two or more replicas.
- **When you enable multi-AZ for a Proxy Cluster instance:**
  - Service running may fluctuate during the change. Perform this operation during off-peak hours.
  - If your application cannot reconnect or handle exceptions, try restarting the application after the change.
- **When you enable multi-AZ for a Redis Cluster instance:**
  - Changing AZs will not interrupt services or the master node, but will slightly affect performance. Perform this operation during off-peak hours.
  - Changing AZs interrupts connections to some replicas. Ensure your application can automatically recover from exceptions and reconnect to Redis.

Procedure

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

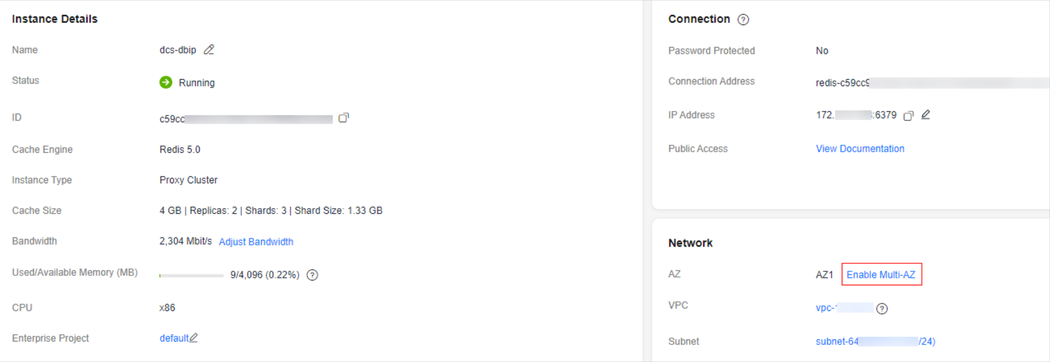
Step 2 Click in the upper left corner of the management console and select the region where your instance is located.

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

Step 4 On the **Cache Manager** page, click a DCS instance.

Step 5 In the **Network** area of the DCS instance, click **Enable Multi-AZ** next to **AZ**.

Figure 7-3 Enabling multi-AZ



Instance Details

Name	dcs-dbip
Status	Running
ID	c59cc[redacted]
Cache Engine	Redis 5.0
Instance Type	Proxy Cluster
Cache Size	4 GB | Replicas: 2 | Shards: 3 | Shard Size: 1.33 GB
Bandwidth	2,304 Mbit/s Adjust Bandwidth
Used/Available Memory (MB)	9/4,096 (0.22%)
CPU	x86
Enterprise Project	default

Connection

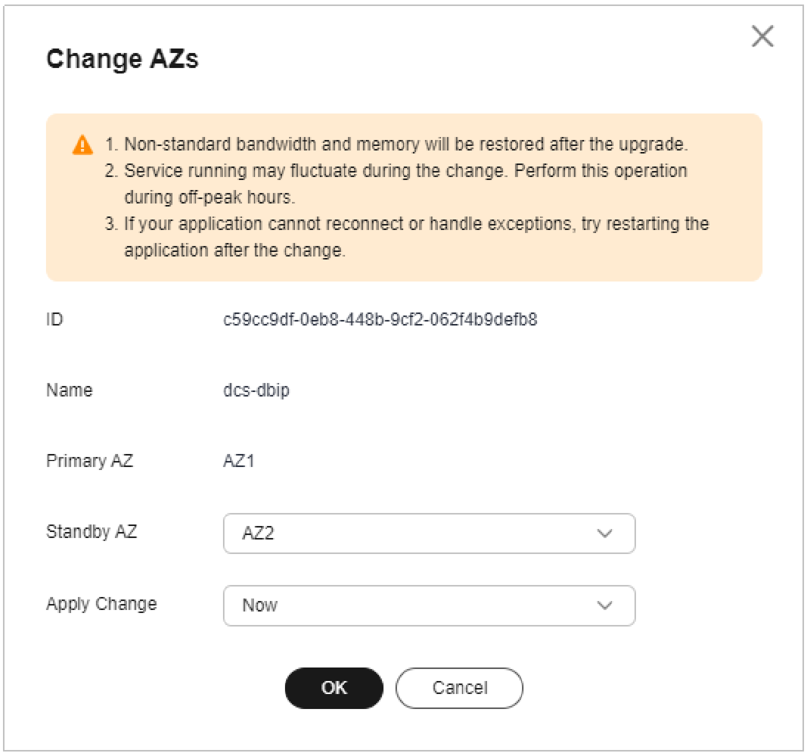
Password Protected	No
Connection Address	redis-c59cc9[redacted]
IP Address	172.[redacted]:6379
Public Access	View Documentation

Network

AZ	AZ1 Enable Multi-AZ
VPC	vpc-[redacted]
Subnet	subnet-64[redacted]/24)

Step 6 In the displayed **Change AZs** dialog box, specify **Standby AZ**.

Figure 7-4 Selecting a standby AZ



Change AZs

1. Non-standard bandwidth and memory will be restored after the upgrade.
2. Service running may fluctuate during the change. Perform this operation during off-peak hours.
3. If your application cannot reconnect or handle exceptions, try restarting the application after the change.

ID	c59cc9df-0eb8-448b-9cf2-062f4b9defb8
Name	dcs-dbip
Primary AZ	AZ1
Standby AZ	AZ2
Apply Change	Now

Step 7 Set **Apply Change** to **Now** or **During maintenance**.

Step 8 Click **OK**.

When complete, the task changes to the **Successful** state.

----End

8 Managing Lifecycle of an Instance

8.1 Restarting a DCS Instance

To recover an instance in cases such as high memory fragmentation ratio or fault occurrence, try restarting the instance on the DCS console. DCS instances can be restarted in batches.

Notes and Constraints

- The DCS instances must be in the **Running** or **Faulty** state.
- For single-node instances or master/standby, cluster, and read/write splitting ones with AOF persistence disabled (parameter **appendonly** set to **no**), the instance data will be cleared after an instance restart. Exercise caution.
- 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 cancels the backup task, or may result in a failure.
- Restarting a DCS instance will disconnect the original client. You are advised to configure automatic reconnection in your application.

Restarting a DCS Instance

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 the names of one or more desired DCS instances and click **Restart** above the list.

To restart one instance, you can also locate the row containing the desired instance, and click **Restart** in the **Operation** column.

Step 5 In the displayed dialog box, click **Yes**. After DCS instances are restarted, their status changes to **Running**.

The time required for restarting a DCS instance depends on the cache size of the instance. It make take **10s to 30 minutes**.

----End

8.2 Starting or Stopping a DCS Instance

Redis 4.0 and later instances support instance stop. When an instance is stopped, data reading or writing is stopped so that the instance cannot be modified, configured, backed up, or migrated. You can neither change the password nor analyze the cache.

NOTE

- A Redis instance is in **Running** state by default, or is in **Stopped** state after you stop it.
- Stopping an instance does not affect its billing.

Starting or Stopping a DCS Instance

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 Stop or start DCS instances.

- Stopping instances
  a. Locate the desired instance and choose **More** > **Stop** in the **Operation** column. You can also select desired instances on the left and choose **More** > **Stop** above.
  b. A dialog box is displayed. Click **Yes**. When the instance is in **Stopped** state, the instance is stopped.
- Starting instances
  a. To restart an instance, click **Start** in the **Operation** column of the desired instance, or select desired instances on the left and choose **More** > **Start** above.
  b. A dialog box is displayed. Click **Yes**. The instance is started when it is in the **Running** state.

----End

8.3 Deleting a DCS 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.

Notes and Constraints

- The DCS instances exist, and must be in the **Running**, **Faulty**, or **Stopped** state.
- Deleting DCS instances removes their data and backups permanently. To retain the backups, download and save them first.
- If the instance is in cluster mode, all cluster nodes will be deleted.
- Instances billed on a yearly/monthly basis cannot be deleted.

Deleting a DCS Instance

Deleting Successfully Created 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 Instances That Failed to Be Created

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

8.4 Clearing DCS Instance Data

To clear instance data, use the **FLUSHDB** or **FLUSHALL** commands on accessed instances, or the data clearance function on the DCS console. This section describes how to use the function to clear instance data with one click.

NOTICE

Clearing instance data may cause the service latency to increase sharply.

Notes and Constraints

- The instances must be of Redis 4.0 and later, and in the **Running** state.
- Clearing instance data cannot be undone and cleared data cannot be recovered. Exercise caution when performing this operation.

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 Data** above the instance list.

Step 6 In the displayed dialog box, click **Yes**.

----End

9 Diagnosing and Analyzing an Instance

9.1 Querying Big Keys and Hot Keys in a DCS Redis Instance

Big keys and hot keys are common issues. This section describes the big key and hot key analysis function on the DCS console. This function monitors the key that occupies most space of a Redis instance, or that is most frequently accessed from the storage data.

- There are two types of big keys:
  - The key value occupies much storage space. If the size of a single String key exceeds 10 KB, or if the size of all elements of a key combined exceeds 50 MB, the key is defined as a big key.
  - The key contains many elements. If the number of elements in a key exceeds 5000, the key is defined as a big key.
- A hot key is most frequently accessed, or consumes significant resources. For example:
  - In a cluster instance, a shard processes 10,000 requests per second, among which 3000 are performed on the same key.
  - In a cluster instance, a shard uses a total of 100 Mbits/s inbound and outbound bandwidth, among which 80 Mbits/s is used by the **HGETALL** operation on a Hash key.

Notes and Constraints

Notes on 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, read/write splitting, 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.

- 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.

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.
- A maximum of 100 hot key 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

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

Querying Big Keys

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.

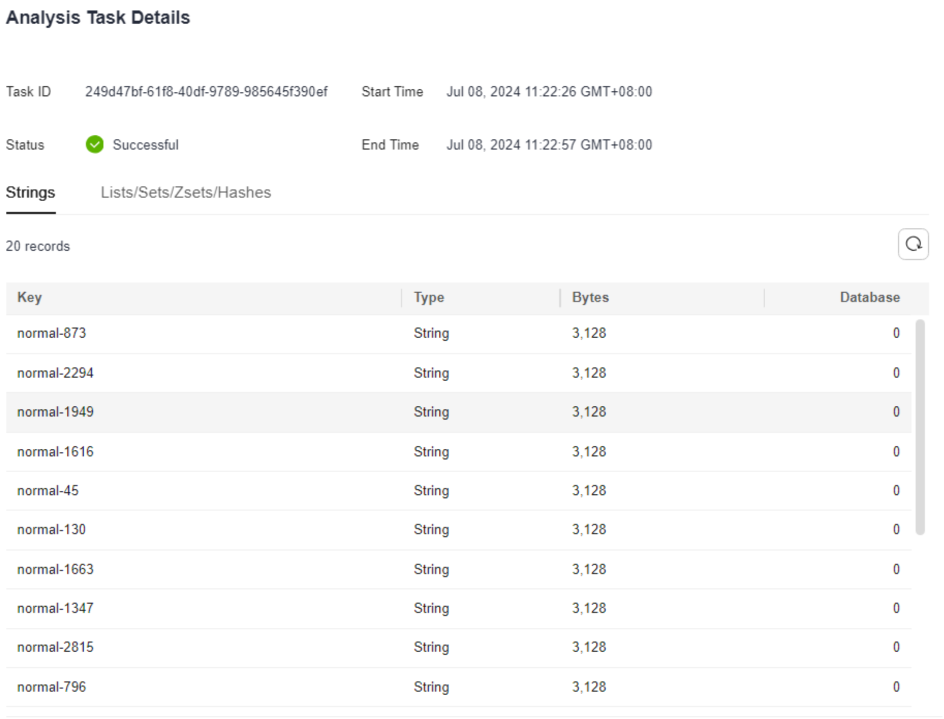
Step 7 After an analysis task completes, click **View** to view the analysis results of different data types.

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

NOTE

The big key analysis result shows records of the top 100 (20 for Strings and 80 for Lists/Sets/Zsets/Hashes) data size.

Figure 9-1 Viewing the results of big key analysis (for Strings)



Analysis Task Details

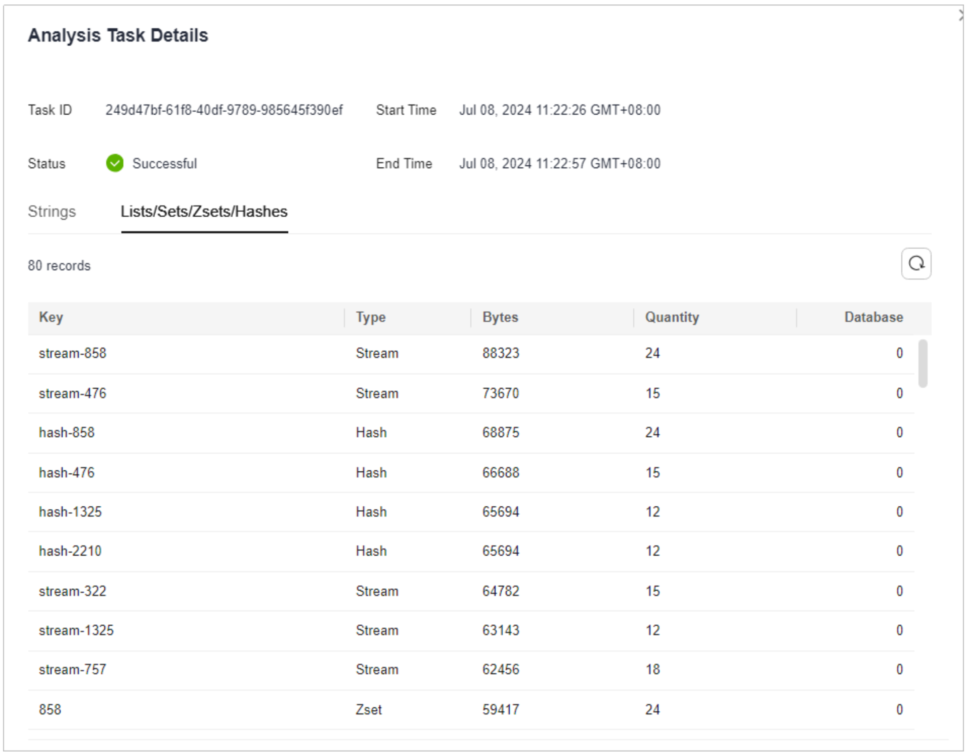
Task ID 249d47bf-61f8-40df-9789-985645f390ef Start Time Jul 08, 2024 11:22:26 GMT+08:00

Status ✓ Successful End Time Jul 08, 2024 11:22:57 GMT+08:00

20 records

Key	Type	Bytes	Database
normal-873	String	3,128	0
normal-2294	String	3,128	0
normal-1949	String	3,128	0
normal-1616	String	3,128	0
normal-45	String	3,128	0
normal-130	String	3,128	0
normal-1663	String	3,128	0
normal-1347	String	3,128	0
normal-2815	String	3,128	0
normal-796	String	3,128	0

Figure 9-2 Viewing the results of big key analysis (for Lists/Sets/Zsets/Hashes)



Analysis Task Details

Task ID 249d47bf-61f8-40df-9789-985645f390ef Start Time Jul 08, 2024 11:22:26 GMT+08:00

Status ✓ Successful End Time Jul 08, 2024 11:22:57 GMT+08:00

80 records

Key	Type	Bytes	Quantity	Database
stream-858	Stream	88323	24	0
stream-476	Stream	73670	15	0
hash-858	Hash	68875	24	0
hash-476	Hash	66688	15	0
hash-1325	Hash	65694	12	0
hash-2210	Hash	65694	12	0
stream-322	Stream	64782	15	0
stream-1325	Stream	63143	12	0
stream-757	Stream	62456	18	0
858	Zset	59417	24	0

Table 9-1 Results of big key analysis

Parameter	Description
Key	The key name in a big key analysis result.
Type	Type of a key, which can be string, hash, list, set, or zset.
Size	The value size of a key, in Bytes.
Quantity	Number of elements in a key. This parameter is displayed only for list, set, zset, or hash type. Unit: counts
Database	Database where the key is located.

----End

Querying Hot Keys

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?**

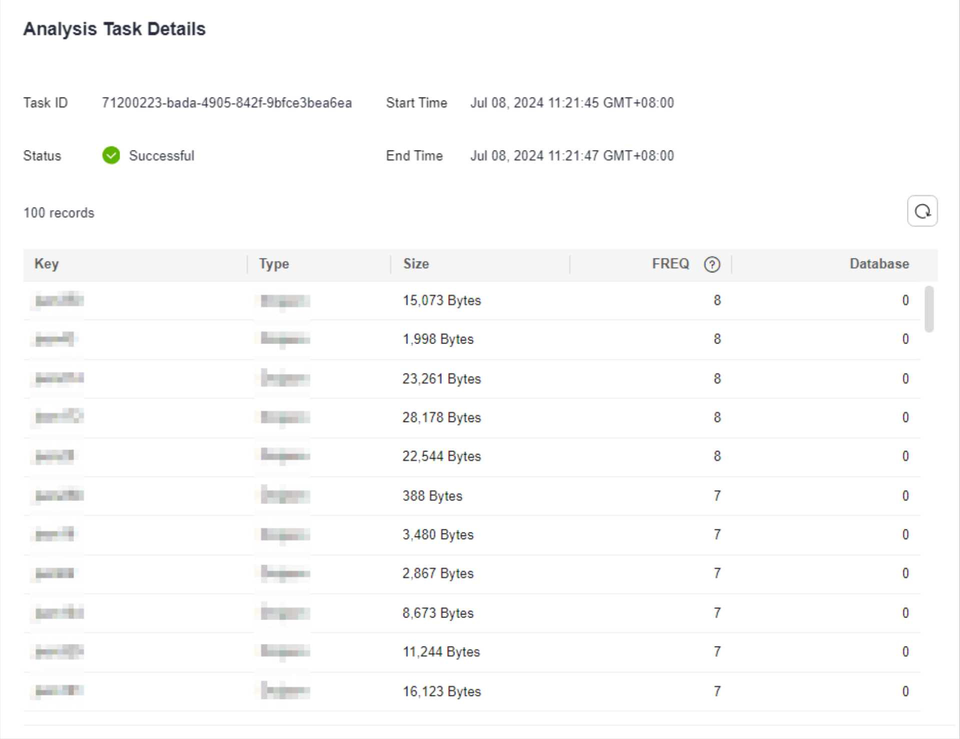
Step 7 After an analysis task completes, click **View** to view the analysis results.

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

NOTE

The hot key analysis result shows the most 100 frequently accessed keys within the specified period.

Figure 9-3 Viewing the results of hot key analysis



Analysis Task Details

Task ID 71200223-bada-4905-842f-9bfce3bea6ea Start Time Jul 08, 2024 11:21:45 GMT+08:00

Status Successful End Time Jul 08, 2024 11:21:47 GMT+08:00

100 records

Key	Type	Size	FREQ	Database
[redacted]	[redacted]	15,073 Bytes	8	0
[redacted]	[redacted]	1,998 Bytes	8	0
[redacted]	[redacted]	23,261 Bytes	8	0
[redacted]	[redacted]	28,178 Bytes	8	0
[redacted]	[redacted]	22,544 Bytes	8	0
[redacted]	[redacted]	388 Bytes	7	0
[redacted]	[redacted]	3,480 Bytes	7	0
[redacted]	[redacted]	2,867 Bytes	7	0
[redacted]	[redacted]	8,673 Bytes	7	0
[redacted]	[redacted]	11,244 Bytes	7	0
[redacted]	[redacted]	16,123 Bytes	7	0

Table 9-2 Results of hot key analysis

Parameter	Description
Key	The key name in a hot key analysis result.
Type	Type of a key, which can be string, hash, list, set, or zset.
Size	The value size of a key, in Bytes.
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 key is located. **NOTE** This parameter is displayed only for cluster instances.
Database	Database where a 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 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?

9.2 Scanning and Deleting Expired Keys in a DCS Redis Instance

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. Instead of checking all keys each time, open-source Redis randomly checks 20 keys each time, 10 times per second by default. This avoids prolonging blocks on the Redis main thread, but the memory used by expired keys cannot be released quickly.

DCS integrates these strategies, and provides a common expired key query method to allow you to periodically release the memory used by expired keys. You can configure scheduled scans on the master nodes of your instances. The entire keyspace 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.

Notes and Constraints

Released expired keys cannot be queried.

Scanning and Deleting Expired Keys in a DCS Redis Instance

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 **Table 9-3** and **Automated Scan Performance and Suggestions**.

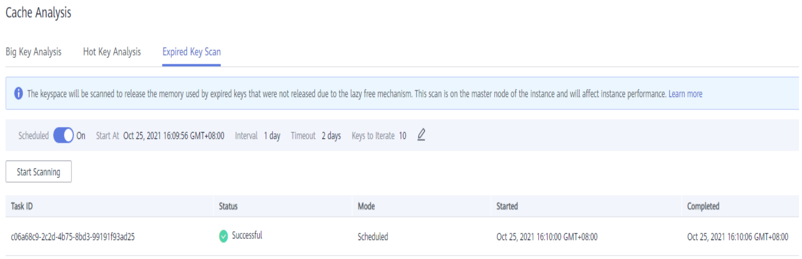
Table 9-3 Parameters for scheduling automatic scans

Parameter	Description
Start At	The first scan can only start after the current time. Format: MM/DD/YYYY hh:mm:ss
Interval	Interval between scans. ● 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. **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 **Automated Scan Performance and Suggestions**. Value range: 0–43,200 Default value: 1440 Unit: minute
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. ● 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. Value range: 1–86,400 Default value: 2880 Unit: minute

Parameter	Description
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. 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. Value range: 10–1,000 Default value: 10 Unit: number

Step 7 After an expired key scan task is submitted, a task record is generated for each expired key scan. You can view the task ID, status, scan mode, start time, and end time.

Figure 9-4 Expired key scan tasks



The keyspace will be scanned to release the memory used by expired keys that were not released due to the lazy free mechanism. This scan is on the master node of the instance and will affect instance performance. Learn more

Scheduled On Start At Oct 25, 2021 16:09:56 GMT+08:00 Interval 1 day Timeout 2 days Keys to Iterate 10

Task ID	Status	Mode	Started	Completed
c06a68c9-2c2d-4b75-8bd3-99191f93ad25	Successful	Scheduled	Oct 25, 2021 16:10:00 GMT+08:00	Oct 25, 2021 16:10:06 GMT+08:00

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.

----End

Automated Scan Performance and Suggestions

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.

9.3 Diagnosing a DCS Redis Instance

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.

Diagnosing a DCS Redis Instance

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.

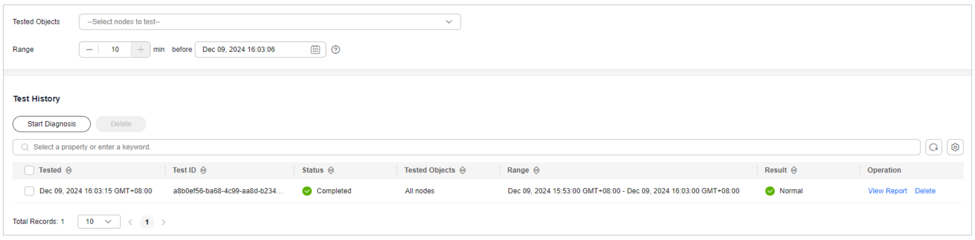
Step 5 Choose **Analysis and Diagnosis** > **Instance Diagnosis**.

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 9-5 Specifying the tested object and time range



Tested Objects: --Select nodes to test--

Range: 10 min before Dec 09, 2024 16:03:06

Test History

Start Diagnosis | Delete

Select a property or enter a keyword.

	Tested	Test ID	Status	Tested Objects	Range	Result	Operation
□	Dec 09, 2024 16:03:15 GMT+08:00	a8b0ef56-ba68-4c99-aa8d-b234...	✓ Completed	All nodes	Dec 09, 2024 15:53:00 GMT+08:00 - Dec 09, 2024 16:03:00 GMT+08:00	✓ Normal	View Report Delete

Total Records: 1 | 10 | 1

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

9.4 Viewing Slow Queries of a DCS Redis Instance

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 number of slow queries in a record. The default value is **128**, which means a maximum of 128 latest slow queries can be displayed.

For details about the configuration parameters, see **Modifying Configuration Parameters of a DCS Instance**.

Notes and Constraints

- Currently, you can view slow queries in the last seven days.
- After restarting an instance, slow queries before the restart cannot be viewed.

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**.

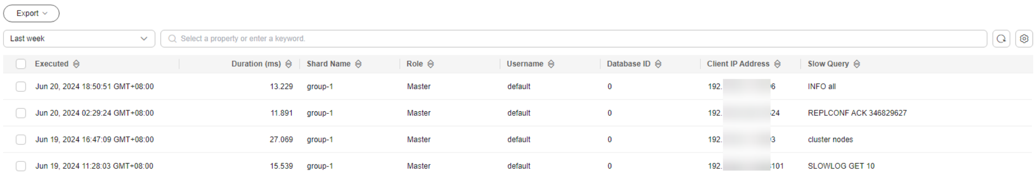
Step 4 Click the name of a DCS instance.

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. For details about the commands, visit the **Redis official website**.

To filter slow queries, click the filtering bar, select a property or enter a keyword.

Figure 9-6 Slow queries of an instance



□ Executed	Duration (ms)	Shard Name	Role	Username	Database ID	Client IP Address	Slow Query
□ Jun 20, 2024 18:50:51 GMT+08:00	13.229	group-1	Master	default	0	192.[redacted]6	INFO all
□ Jun 20, 2024 02:29:24 GMT+08:00	11.891	group-1	Master	default	0	192.[redacted]24	REPLCONF ACK 346829627
□ Jun 19, 2024 16:47:09 GMT+08:00	27.069	group-1	Master	default	0	192.[redacted]3	cluster nodes
□ Jun 19, 2024 11:28:03 GMT+08:00	15.539	group-1	Master	default	0	192.[redacted]101	SLOWLOG GET 10

Step 7 To download slow queries, choose **Export** > **Export all data to an XLSX file** or **Export selected data to an XLSX file**.

----End

9.5 Viewing Redis Run Logs

Run logs of a Redis instance can be queried on the DCS console. Logs of a specified time can be collected into the **redis.log** file, and downloaded to the local.

Instance running exceptions include AOF rewrites, configuration modifications, critical operations, and master/standby switchovers.

Notes and Constraints

- The logs are retained for seven days, and are automatically deleted later.
- A maximum of seven days of run logs can be queried for a Redis instance.

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 **Collect Logs**, specify the collection period, and click **OK**.

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.

A log file contains logs of one day. For example, if you select last 3 days, three log files will be generated.

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

10 Migrating Instance Data

10.1 DCS Data Migration Overview

The DCS console supports online and backup (file) migration with intuitive operations. Incremental data can be migrated online.

- 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**.
- 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.

Users can customize migration solutions as required based on specific Redis environment or scenarios. The data volume, source Redis deployment, and network bandwidth affect migration duration. The actual duration depends.

Before migrating an instance, analyze the cache commands (reference: **Command Compatibility**) used by your service systems and verify the commands one by one during the drill phase. Contact customer service as needed.

NOTICE

- Currently, the data migration function is free of charge in the OBT. You will be notified when data migration starts to be charged.
- As an important and stringent task, data migration requires high accuracy and timeliness, which depends on specific services and operations.
- Cases provided in this document are for reference only. Consider your needs during actual migration.
- Some commands in this document contain the instance password, which means the passwords are recorded. Ensure that the passwords are not disclosed and clear operation records in a timely manner.

DCS Data Migration Modes

NOTE

- **DCS for Redis** refers to Redis instances provided by Huawei Cloud DCS.
- **Self-hosted Redis** refers to self-hosted Redis on the cloud, from other cloud vendors, or in on-premises data centers.
- **√**: Supported. **×**: Not supported.
- 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 more information, see **Migration Solution Notes**.

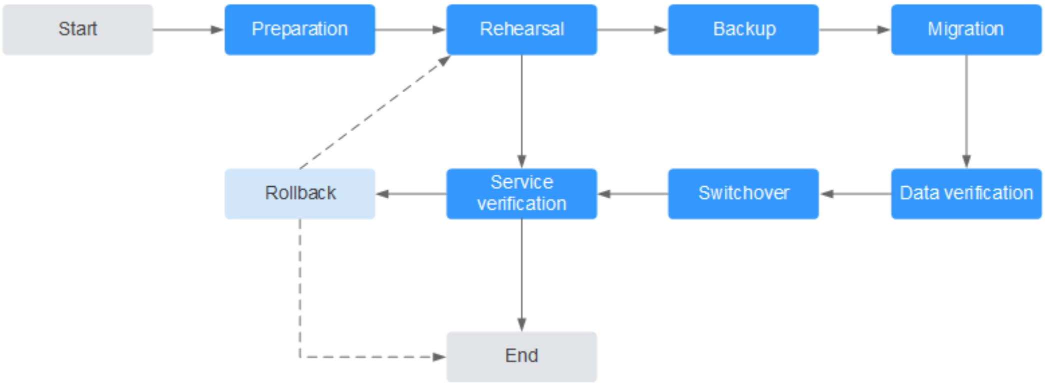
Table 10-1 DCS data migration modes

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	√	√	√

Migration Process

Figure 10-1 Migration flowchart



1. Evaluation

   Collect the following information about the cached data to be migrated (based on **Table 10-2**):

   – Number of instances

   – Number of databases (DBs) configured for each instance

   – Number of keys in each DB

   – DBs used for your services

   – Space occupied by each instance

   – Redis version

   – Redis instance type

   – Mapping relationships between your services and instances

   NOTE

   - The **info keyspace** command can be used on accessed source Redis to check whether databases have available data and the key quantity in databases. The key quantity in each database can be used for migration verification.
   - The **info memory** command can be used on accessed source Redis to check the source Redis data volume by the returned **used_memory_human** value. Check whether the following resources are sufficient: available ECS space, target instance flavor, and remaining memory (≥ source Redis data).

   Plan the following information about target DCS instances based on the collected information:

   – Number of instances to be applied for

   – Memory size of the DCS instance (≥ that of source Redis)

   – Version of the DCS instance (≥ that of source Redis)

   – DCS instance type

   - Virtual Private Clouds (VPCs) and subnets, and security groups, to which the instances and services belong

2. Preparation

   After completing the evaluation, prepare the following items:

   a. Mobile storage devices

      These devices are used to copy and transfer data in case of network disconnection (in scenarios with data centers of enterprises).

   b. Network resources

      Create VPCs and subnets based on service planning.

   c. Server resources

      Apply for ECSs to bear Redis clients. The ECSs are used to export or import cached data.

      Recommended ECS specifications are 8 vCPUs | 16 GB or higher.

   d. DCS instances

      Create DCS instances based on the migration planning. If the number of instances exceeds the default quota, submit a service ticket or contact customer service.

   e. Related tools

      Install the FTP tool.

   f. Information to be collected

      Collect the contact information of people involved in the migration, ECS login credentials, cache instance information, and DB information.

   g. Overall migration plan

      Formulate the overall migration plan, including the personnel arrangement, drill, migration, verification, service switchover, and rollback solutions.

      Break down each solution into executable operations and set milestones to mark the end of tasks.

3. Drill

   The drill phase aims to:

   a. Verify the feasibility of the migration tools and migration process.
   b. Discover problems that may occur during migration and make effective improvements.
   c. Evaluate the time required for migration.
   d. Optimize the migration steps and verify the feasibility of concurrent implementation of some tasks to improve migration efficiency.

4. Backup

   Before migration, back up related data, including but not limited to cached data and Redis configuration files, in case of emergency.

5. Migration

   After conducting one or two rounds of migration drill and solving problems found in the drill, start data migration.

   Break down the migration process into executable steps with specific start and end confirmation actions.

6. Data verification

   Check the following items:

   - The key distribution of each DB is consistent with the original or expected distribution.
   - Main keys.
   - Expiration time of keys.
   - Whether instances can be normally backed up and restored.

7. Service switchover

   a. After the data migration and verification, use the new instances for your services.
   b. If DB IDs are changed, modify the ID configurations for your services.
   c. If your services are migrated from data centers or cloud platforms provided by other vendors to Huawei Cloud as a whole, services and cached data can be migrated concurrently.

8. Service verification

   a. Verify the connectivity between your service applications and DCS instances.
   b. Verify whether cached data can be normally added, deleted, modified, and queried.
   c. If possible, perform pressure tests to ensure that the performance satisfies the peak service pressure.

9. Rollback

   If your services are unavailable after the data migration because unexpected problems occur and cannot be solved in the short term, roll back your services.

   Since source Redis data still exists, you only need to roll back your services and use the source Redis instances again.

   After the rollback, you can continue to restart from the drill or even preparation phase to solve the problems.

Information to be collected for the migration

The following table lists the information to be collected in the evaluation and preparation phases.

Table 10-2 Information to be collected for the migration

Migration Source	Item	Description
Source Redis (List the information about all instances to be migrated.)	Source Redis IP address	-
	Redis instance password (if any)	-
	Total data volume	Run the **info memory** command and refer to the **used_memory_human** value to obtain the total data volume. Used to evaluate whether the migration solution, DCS instance specifications, and available disk space of ECSs meet requirements, and to estimate the time required for migration (service interruption duration).
	IDs of DBs with data	Obtained by running the **info keyspace** command. Used to check whether the migration involves multiple DBs and non-AOF files. Some open-source tools can export and import data of only one DB at a time. For DCS instances, the single-node and master/standby types provide 256 DBs (DB 0 to DB 255), and the cluster type provides only one DB by default.
	Number of keys in each DB	Used to verify the data integrity after migration.
	Data type	The Cloud Data Migration (CDM) service supports two data formats: hash and string. If the source data contains data in other formats such as list and set, use a third-party migration tool.
Huawei Cloud ECS If a large number of instances are to be migrated, prepare multiple ECSs for concurrent migration.	EIP	Select ECSs that can communicate with DCS instances for data import to ensure network stability. Configure high-specification bandwidth to improve data transmission efficiency.
	Login credentials (username and password)	-

Migration Source	Item	Description
	CPU and memory	Some migration tools support concurrent import through multiple threads. High-specification ECSs help improve import efficiency.
	Available disk space	Sufficient available disk space needs to be reserved on the ECSs to store compressed files and decompressed cached data files. Note: To improve data transmission efficiency, compress large-size data files before transmitting them to ECSs.
DCS instances (Select appropriate instance specifications and quantities based on the number of source Redis instances and data volume.)	Instance connection address	-
	Instance connection port	-
	Instance password	-
	Instance type	-
	Instance specifications and available memory	-
Network configurations	VPC	Plan VPCs in advance to ensure that your service applications and DCS instances are in same VPCs.
	Subnet	-
	Whitelist	For details, see **Managing IP Address Whitelist**.
-	-	*Other configurations*.

10.2 Migration Solution Notes

Migration Tools

Table 10-3 Comparing Redis migration tools

Tool/ Command/ Service	Feature	Description
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. ● 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.
redis-cli	● The Redis command line interface (CLI), which can be used to export data as an RDB file or import the AOF file (that is, all DBs) of an instance. ● An AOF file is large file containing a full set of data change commands.	-
Rump	Supports online migration between DBs of an instance or between DBs of different instances.	Rump does not support incremental migration. Stop services before migrating data. Otherwise, keys might be lost. For details, see **Online Migration from Another Cloud Using Rump**.

Tool/ Command/ Service	Feature	Description
Redis-shake	An open-source tool that supports both online and offline migration.	redis-shake is suitable for migrating Redis Cluster data.
Self-developed migration script	Flexible and can be adjusted as required.	-

Migration Schemes

Table 10-4 Migration Schemes

Scenario	Tool	Use Case	Description
Migration between Huawei Cloud DCS instances	DCS console	- To migrate instances in a region and of an account, see **Online Migration Between Instances**. - To migrate instances in different regions or accounts, see **Backup Import Between DCS Redis Instances**.	**Attempts to migrate data from a later-version Redis instance to an earlier-version Redis instance are not recommended because they will fail** due to data compatibility issues between different Redis versions.

Scenario	Tool	Use Case	Description
From self-hosted Redis to DCS **NOTE** **Self-hosted Redis** refers to self-hosted Redis on Huawei Cloud, in another cloud, or in on-premises data centers.	DCS console	- If the network between your self-hosted Redis instance and the DCS Redis instance is connected, follow to the instructions in **Online Migration Between Instances**. - If the network between your self-hosted Redis instance and the DCS Redis instance is not connected, follow to the instructions in **Self-Hosted Redis Migration with Backup Files**.	-
	redis-cli	- **Self-Hosted Redis Migration with redis-cli (AOF)** - **Self-Hosted Redis Migration with redis-cli (RDB)**	-
	redis-shake	- **Self-Hosted Redis Cluster Migration with redis-shake (Online)** - **Self-Hosted Redis Cluster Migration with redis-shake (RDB)**	-
From another cloud to DCS	DCS console	- If the **SYNC** and **PSYNC** commands are not disabled for the Redis service provided by another cloud, follow the instructions in **Migrating Redis from Another Cloud Online**. - If the **SYNC** and **PSYNC** commands are disabled for the Redis service provided by another cloud, follow the instructions in **Backup Import from Another Cloud**.	If online migration is required, contact the O&M personnel of another cloud to enable the **SYNC** and **PSYNC** commands.
	Rump	**Online Migration from Another Cloud Using Rump**	-

Scenario	Tool	Use Case	Description
	Redis-shake	**Backup Import from Another Cloud Using redis-shake** **Migrating from Another Cloud Online Using redis-shake**	-

10.3 Migrating Data Between DCS Instances

10.3.1 Online Migration Between Instances

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.

NOTE

During online migration, data is essentially synchronized in full to a new replica. Therefore, perform online migration during low-demand hours. Otherwise, source instance CPU usage may surge and latency may increase.

Notes and Constraints

- You cannot use public networks for online migration.
- Before migrating data, read through **Migration Tools and Schemes** to learn about the DCS data migration function and select an appropriate target instance.
- Migrating a later Redis instance to an earlier one may fail.
- For earlier instances whose passwords contain single quotation marks ('), modify the password for online migration or try other methods.
- By default, a Proxy Cluster instance has only one database (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to **Enabling Multi-DB**.
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to **Online Migration with Rump**.
- 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.

Prerequisites

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**.

  If the data exists on the target instance, the replicated data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Creating an Online Migration Task

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

If the source and target Redis are under different accounts, use the source account to log in to DCS.

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

NOTE

Only when the online migration task and the source Redis are under an account and in a region, the **SYNC** and **PSYNC** commands of the source Redis are allowed. Otherwise, online migration cannot be performed.

Step 3 In the navigation pane, choose **Data Migration**. The migration task list is displayed.

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

Step 5 Enter the task name and description.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

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

NOTE

- Use the VPC of the source or target Redis.
- The online 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

Checking 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 **Configuring the 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 **Configuring the 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 DCS region, create a VPC peering connection by referring to **VPC Peering Connection**.

----End

Configuring the Online Migration Task

Step 1 On the **Online Migration** tab page, click **Configure** in the row containing the online migration task you just created.

Step 2 Select a migration type.

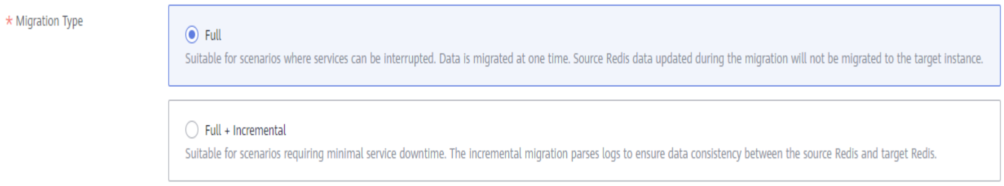
Supported migration types are **Full** and **Full + Incremental**, which are described in **Table 10-5**.

To **switch DCS instance IPs** after instance migration, select **Full + Incremental** for the migration type.

Table 10-5 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 10-2 Selecting the migration type



* Migration Type

● Full
Suitable for scenarios where services can be interrupted. Data is migrated at one time. Source Redis 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 Redis and target Redis.

Step 3 Only 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**.

1. Set **Source Redis Type** to **Redis in the cloud** and add **Source Redis Instance**.
2. Configure **Target Redis Type** and **Target Redis Instance**:
   - If the target Redis and migration task are in a VPC, or across VPCs over a network in a region, set **Target Redis Type** to **Redis in the cloud** and add **Target Redis Instance**.
   - If the target Redis and migration task are in different regions, set **Target Redis Type** to **Self-hosted Redis** and add **Target Redis Instance**. If the target Redis is a Redis Cluster, enter the IP addresses and ports of all masters in the cluster and separate multiple addresses with commas (,). For example: **192.168.1.1:6379,192.168.0.0:6379**
3. Configure **Source Redis Instance Password** and **Target Redis Instance Password**: 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.

   Currently, the users created in **Managing Users** are unavailable here.
4. You can specify the source DB and target DB. For example, if you enter **5** for source DB and **6** for target DB, data in DB5 of the source Redis will be migrated to DB6 of the target Redis. If the source DB is not specified but the target DB is specified, all source data will be migrated to the specified target DB by default. If the target DB is not specified, data will be migrated to the corresponding target DB.

   NOTE

   - If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
   - For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

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**.

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

NOTE

- Once full + incremental migration starts, it remains **Migrating** after full migration.
- To manually stop a migration task, select the check box on the left of the migration task and click **Stop** above the migration task.
- To perform migration again, select the migration tasks which failed or are stopped, and click **Restart** above. If a restarted migration task fails, click **Configure** to configure the task and try again.
- A maximum of 50 online migration tasks can be selected at a time. You can stop, delete, or restart them in batches.

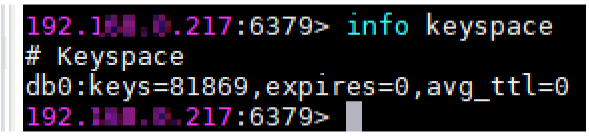
----End

Verifying the Migration

After the migration is complete, check data integrity in the following way.

1. Connect the source Redis and the target Redis. For details, see **redis-cli**.
2. Run the **info keyspace** command on the source and the target Redis to check the values of **keys** and **expires**.

Figure 10-3 Checking instance data



```
192.1[redacted].217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.[redacted].217:6379>
```

3. Calculate the differences 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.

NOTE

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

(Optional) Switching DCS Instance IP Addresses

The prerequisites for switching source and target Redis instance IP addresses are as follows. The target Redis can be accessed automatically on a client after the switch.

Prerequisites:

- This function is supported by basic edition DCS Redis 4.0 instances and later, **but not by professional edition DCS Redis instances**.
- For DCS Redis 3.0 instances, contact customer service to enable the whitelist for Redis 3.0 instance IP switches. The instance IP addresses can be switched only when the source instance is a DCS Redis 3.0 instance and the target instance is a basic edition DCS Redis 4.0, 5.0, or 6.0 instance.
- The IP addresses of a source or target instance with public access enabled cannot be switched.
- Instance IPs can be switched only for the source and target Redis that are single-node, master/standby, read/write splitting, or Proxy Cluster instances.

- **Full + Incremental** must be selected in **Step 2**.
- The source and target Redis instance ports must be consistent.

NOTICE

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. If your application cannot reconnect to Redis or handle exceptions, you may need to restart the application after the IP switching.
4. If the source and target instances are in different subnets, the subnet information will be updated after the switching.
5. 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.
6. 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**.
7. Ensure that the passwords of the source and target instances are the same. If they are different, verification will fail after the switching.
8. If a whitelist is configured for the source instance, ensure that the same whitelist is configured for the target instance before switching IP addresses.

Step 1 On the **Data Migration** > **Online Migration** page, when the migration task status changes to **Incremental migration in progress**, choose **More** > **Switch IP** in the **Operation** column.

Step 2 In the **Switch IP** dialog box, select whether to switch the domain name.

NOTE

- If a Redis domain name is used on the client, switch it or you must modify the domain name on the client.
- If the domain name switch is not selected, only the instance IP addresses will be switched.

Step 3 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.

To restore the IPs, choose **More** > **Roll Back IP** in the operation column. The IPs are rolled back when the task is in the **Successful** state.

----End

10.3.2 Backup Import Between DCS Redis Instances

You can migrate data between DCS instances by importing backup files.

- If the source Redis and target Redis are in the same region under the same DCS account, and the source Redis is not a single-node instance, see **Importing Backup Data from a Redis Instance**.

- If the source Redis and target Redis are in different regions or under different DCS accounts, or the source Redis is a single-node instance, see **Importing Backup Data from an OBS Bucket**.

Prerequisites

- You have successfully backed up the source Redis instance.
  - For **Importing Backup Data from a Redis Instance**, you do not need to download the backup file to the local PC. For details about how to back up data, see **Manually Backing Up a DCS Instance**.
  - For **Importing Backup Data from an OBS Bucket**, download the backup file to the local PC by referring to **Downloading a Backup File**.
- You have prepared the target Redis instance. If a target DCS Redis instance is not available, create one first. For details, see **Buying a DCS Redis Instance**.

  Redis is backward compatible. The target instance version must be the same as or later than the source instance version.
- Ensure that the target Redis instance has sufficient storage space. You can clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Importing Backup Data from a Redis Instance

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 source and target instances are located.

Step 3 In the navigation pane, choose **Data Migration**. The migration task list is displayed.

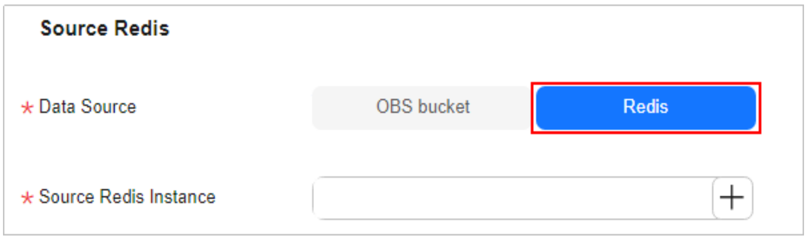
Step 4 Click **Create Backup Import Task**.

Step 5 Enter the task name and description.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

Step 6 For source Redis, set **Data Source** to **Redis**.

Figure 10-4 Selecting a data source (Redis)



Source Redis

* Data Source: OBS bucket | Redis

* Source Redis Instance

Step 7 For **Source Redis Instance**, select the source instance to be migrated.

Step 8 Select the backup task whose data is to be migrated.

Step 9 For **Target Redis Instance**, select the DCS Redis instance prepared in **Prerequisites**.

Step 10 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 11 For **Target DB**, you can specify a DB in the target Redis to migrate data to. For example, if you enter **5**, data will be migrated to DB5 of the target Redis. If you do not specify a DB, data will be migrated to a DB corresponding to the source DB.

NOTE

- If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
- For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

Step 12 Click **Next**.

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

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

----End

Importing Backup Data from an OBS Bucket

Simply download the source Redis data and then upload the data to an OBS bucket in the same account and region as the target DCS Redis instance. After you have created a backup import task, data in the OBS bucket will be read and migrated to the target Redis.

NOTE

- .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.
- To migrate data from a cluster Redis instance, download all backup files and upload all of them to the OBS bucket. Each backup file contains data for a shard of the instance. During the migration, you need to select backup files of all shards.

Step 1 **Create an OBS bucket in the account and region where the target Redis instance is located.** If a qualified OBS bucket is available, you do not need to create one.

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**.

- **Region**:

  The OBS bucket must be in the same region as the target DCS Redis instance.
- **Default Storage Class**: Select **Standard** or **Infrequent Access**.

  Do not select **Archive**. Otherwise, the migration will fail.

Step 2 Upload the backup file to the OBS bucket.

1. In the bucket list, click the name of the created bucket.
2. In the navigation pane, choose **Objects**.
3. On the **Objects** tab page, click **Upload Object**.
4. Specify **Storage Class**.

   Do not select **Archive**. Otherwise, the migration will fail.
5. 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.
6. Specify **Server-Side Encryption**.
7. Click **Upload**.

Step 3 Click ☰ in the upper left corner and choose **Distributed Cache Service for Redis** under **Middleware** to open the DCS console.

Step 4 In the navigation pane, choose **Data Migration**.

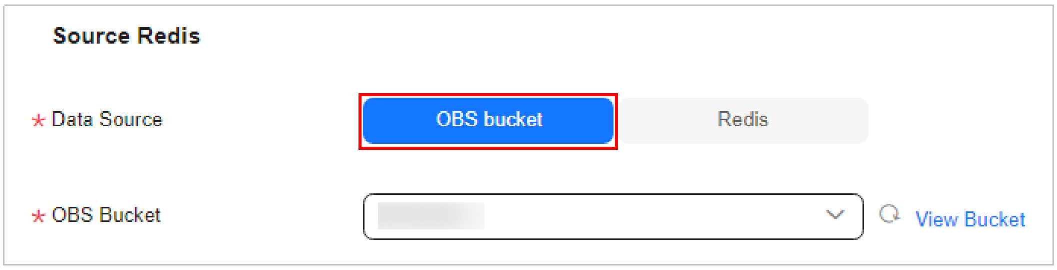
Step 5 Click **Create Backup Import Task**.

Step 6 Enter the task name and description.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

Step 7 In the **Source Redis** area, select **OBS Bucket** for **Data Source** and then select the OBS bucket to which you have uploaded backup files.

Figure 10-5 Selecting a data source (OBS bucket)



Source Redis

* Data Source: OBS bucket | Redis

* OBS Bucket: View Bucket

Step 8 Click **Add Backup** and select the backup files to be migrated.

Step 9 In the **Target Redis** area, select the **Target Redis Instance** prepared in **Prerequisites**.

Step 10 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 11 For **Target DB**, you can specify a DB in the target Redis to migrate data to. For example, if you enter **5**, data will be migrated to DB5 of the target Redis. If you do not specify a DB, data will be migrated to a DB corresponding to the source DB.

NOTE

- If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
- For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

Step 12 Click **Next**.

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

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

----End

10.4 Migrating Data from Self-Hosted Redis to DCS

10.4.1 Migrating Self-Built Redis Online

If the source self-host 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 DCS.

NOTE

During online migration, data is essentially synchronized in full to a new replica. Therefore, perform online migration during low-demand hours. Otherwise, source instance CPU usage may surge and latency may increase.

Notes and Constraints

- If the **SYNC** and **PSYNC** commands are disabled by the source instance, enable them before migrating data. Otherwise, the migration fails.
- 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.
- The source must be Redis 3.0 or later.
- For earlier instances whose passwords contain single quotation marks ('), modify the password for online migration or try other methods.

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 multi-DB single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to **Enabling Multi-DB**.

- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to **Online Migration with Rump**.
- The IP address and port of the source Redis instance has been obtained.
- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Creating an Online Migration Task

Step 1 Log in to the DCS console using the account of the target DCS Redis instance.

Step 2 Click 📍 in the upper left corner of the console and select the region where your target 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.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

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

NOTE

- **Select the same VPC as the target Redis. Ensure that the migration resource can access the target Redis instance.**
- The online 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

Checking 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 **Creating 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 **Configuring the Online Migration Task**. If no, go to **Step 3**.

Step 3 Perform the following operations to establish the network.

- If the VPC of the source and target Redis instances are of the same cloud vendor and 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 using only Direct Connect. For details, see **Direct Connect documentation**.

----End

Configuring the Online Migration Task

Step 1 On the **Online Migration** tab page, click **Configure** in the row containing the online migration task you just created.

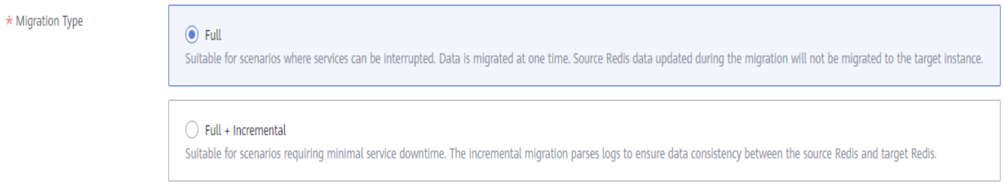
Step 2 Select a migration type.

Supported migration types are **Full** and **Full + Incremental**, which are described in **Table 10-6**.

Table 10-6 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 10-6 Selecting the migration type



* Migration Type

● Full

Suitable for scenarios where services can be interrupted. Data is migrated at one time. Source Redis 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 Redis and target Redis.

Step 3 Only 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**.

1. Configure **Source Redis Type** and **Source Redis Instance**:

   Set **Redis in the cloud** for **Source Redis Type** and add **Source Redis Instance**.

   If the source Redis is a Redis Cluster, enter the IP addresses and ports of all masters in the cluster and separate multiple addresses with commas (,). For example: **192.168.1.1:6379,192.168.0.0:6379**

2. Configure **Target Redis Type** and **Target Redis Instance**:

   Set **Redis in the cloud** for **Target Redis Type** and add **Target Redis Instance**.

3. Configure **Source Redis Instance Password** and **Target Redis Instance Password**: 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. If the test fails, check whether the password is correct, and whether the migration task network is connected.

   If a DCS Redis instance is used, the users created in **Managing Users** are currently unavailable.

4. You can specify the source DB and target DB. For example, if you enter **5** for source DB and **6** for target DB, data in DB5 of the source Redis will be migrated to DB6 of the target Redis. If the source DB is not specified but the target DB is specified, all source data will be migrated to the specified target DB by default. If the target DB is not specified, data will be migrated to the corresponding target DB.

   NOTE

   - If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
   - For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

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**.

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

NOTE

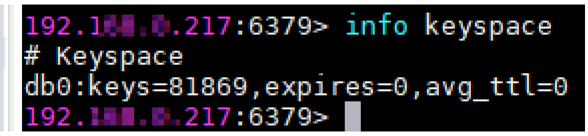
- Once incremental migration starts, it remains **Migrating** after full migration.
- To manually stop a migration task, select the check box on the left of the migration task and click **Stop** above the migration task.

----End

Verifying the Migration

Before data migration, if the target Redis has no data, check data integrity after the migration is complete in the following way:

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



```
192.1[redacted].[redacted].217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.[redacted].[redacted].217:6379>
```

3. Calculate the differences 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.

10.4.2 Self-Hosted Redis Migration with Backup Files

This section describes how to migrate self-hosted Redis to DCS by importing backup files.

Simply download the source Redis data and then upload the data to an OBS bucket in the same Huawei Cloud account and 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.

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 multi-DB single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to **Enabling Multi-DB**.
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to **Online Migration with Rump**.
- Prepare the source Redis backup file. The backup file must be in .aof, .rdb, .zip, or .tar.gz format.

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Creating an OBS Bucket and Uploading Backup Files

If the source Redis backup file to be uploaded is smaller than 5 GB, perform the following steps to create an OBS bucket and upload the file on the OBS console. If the backup file to be uploaded is larger than 5 GB, upload the file by referring to **instructions**.

Step 1 Create an OBS bucket on the OBS console.

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*.

1. **Region**:

   The OBS bucket must be in the same region as the target DCS Redis instance.

2. **Storage Class**: Available options are **Standard**, **Infrequent Access**, and **Archive**.

   Do not select **Archive**. Otherwise, the migration will fail.

Step 2 In the bucket list, click the bucket created in **Step 1**.

Step 3 In the navigation pane, choose **Objects**.

Step 4 On the **Objects** tab page, click **Upload Object**.

Step 5 Specify **Storage Class**.

Do not select **Archive**. Otherwise, the migration will fail.

Step 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.

Step 7 (Optional) Select **KMS encryption** to encrypt the uploaded files.

Step 8 Click **Upload**.

----End

Creating a Migration Task

Step 1 Go to the DCS console.

Step 2 In the navigation pane, choose **Data Migration**.

Step 3 Click **Create Backup Import Task**.

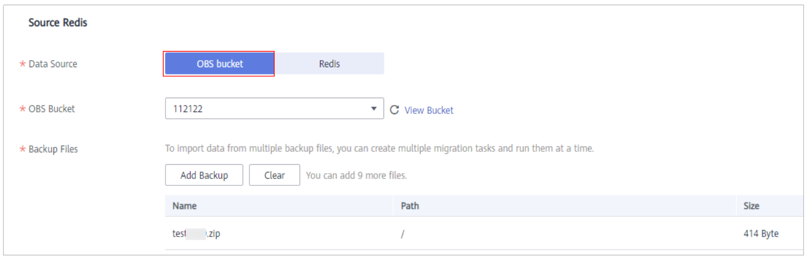
Step 4 Enter the task name and description.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

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.

Figure 10-7 Specifying the backup file information



Source Redis

* Data Source: OBS bucket

* OBS Bucket: 112122

* Backup Files

To import data from multiple backup files, you can create multiple migration tasks and run them at a time.

You can add 9 more files.

Name	Path	Size
test[redacted].zip	/	414 Byte

Step 7 In the **Target Redis** area, select the **Target Redis Instance** prepared in **Prerequisites**.

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 For **Target DB**, you can specify a DB in the target Redis to migrate data to. For example, if you enter **5**, data will be migrated to DB5 of the target Redis. If you do not specify a DB, data will be migrated to a DB corresponding to the source DB.

NOTE

- If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
- For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

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

Verifying the Migration

After the data is imported successfully, access the DCS instance and run the **info** command to check whether the data has been successfully imported as required. Connect to Redis by referring to **redis-cli**.

If the import fails, check the procedure. If the import command is incorrect, run the **flushall** or **flushdb** command to clear the cache data in the target instance, modify the import command, and try again.

10.4.3 Self-Hosted Redis Migration with redis-cli (AOF)

redis-cli is the command line tool of Redis, which can be used after you install the Redis server. This section describes how to use redis-cli to migrate a data from a self-hosted Redis instance to a DCS instance.

An AOF file can be generated quickly. It applies to scenarios where you can access the Redis server and modify the configurations, such as scenarios with self-built Redis servers.

NOTE

- Migrate data during off-peak hours.
- Before data migration, suspend your services so that data changes newly generated will not be lost during the migration.

Prerequisites

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.
- An Elastic Cloud Server (ECS) has been created. For details about how to create an ECS, see **Purchasing an ECS**.

Generating an AOF File

1. Log in to the ECS.
2. Install redis-cli. The following steps assume that your client is installed on the Linux OS.
   a. Run the following command to download Redis: You can also install other Redis versions. For details, see the **Redis official website**.
   ```
   wget http://download.redis.io/releases/redis-5.0.8.tar.gz
   ```
   b. Run the following command to decompress the source code package of your Redis client:
   ```
   tar -xzf redis-5.0.8.tar.gz
   ```
   c. Run the following commands to go to the Redis directory and compile the source code of your Redis client:
   ```
   cd redis-5.0.8
   make
   cd src
   ```
3. Run the following command to enable cache persistence and obtain the AOF persistence file:
   ```
   redis-cli -h {source_redis_address} -p {port} -a {password}  config set appendonly yes
   ```
   {source_redis_address} is the connection address of the source Redis, *{port}* is the port of the source Redis, and *{password}* is the connection password of the source Redis.

   If the size of the AOF file does not change after you have enabled persistence, the AOF file contains full cached data.

NOTE

- To find out the path for storing the AOF file, use redis-cli to access the Redis instance, and run the **config get dir** command. Unless otherwise specified, the file is named as **appendonly.aof** by default.
- To disable synchronization after the AOF file is generated, use redis-cli to log in to the Redis instance and run the **config set appendonly no** command.

Uploading the AOF file to Huawei Cloud ECS

To save time, you are advised to compress the AOF file and upload it to Huawei Cloud ECS using an appropriate mode (for example, SFTP mode).

NOTE

Ensure that the ECS has sufficient disk space for data file decompression, and can communicate with the DCS instance. Generally, the ECS and DCS instance are configured to belong to the same VPC and subnet, and the configured security group rules do not restrict access ports.

Importing Data

Log in to the ECS and run the following command to import data.

```
redis-cli -h {dcs_instance_address} -p {port} -a {password} --pipe < appendonly.aof
```

{dcs_instance_address} indicates the address of the target Redis instance, *{port}* indicates the port of the target Redis instance, and *{password}* indicates the password for connecting to the target Redis instance.

It takes 4 to 10 seconds to import an AOF file of 1 million data (20 bytes per data segment) to a VPC.

Verifying the Migration

After the data is imported successfully, access the DCS instance and run the **info** command to check whether the data has been successfully imported as required. Connect to Redis by referring to **redis-cli**.

If the import fails, check the procedure. If the import command is incorrect, run the **flushall** or **flushdb** command to clear the cache data in the target instance, modify the import command, and try again.

10.4.4 Self-Hosted Redis Migration with redis-cli (RDB)

redis-cli is the command line tool of Redis, which can be used after you install the Redis server. redis-cli supports data export as an RDB file. If your Redis service does not support AOF file export, use redis-cli to obtain an RDB file. Then, use another tool (such as redis-shake) to import the file to a DCS instance.

NOTE

Migrate data during off-peak hours.

Notes and Constraints

- When the source is Redis native cluster data, individually export the data of each node in the cluster, and then import the data node by node.

Prerequisites

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.
- An Elastic Cloud Server (ECS) has been created. For details about how to create an ECS, see **Purchasing an ECS**.
- **The source self-hosted Redis instance must support the SYNC command.** Otherwise, the RDB file cannot be exported using redis-cli.

Exporting the RDB File

1. Prepare for the export.

   For master/standby or cluster DCS instances, there is a delay in writing data into an RDB file based on the delay policies configured in the **redis.conf** file. Before data export, learn the RDB policy configurations of the Redis instance to be migrated, suspend your service systems, and then write the required number of test data into the Redis instance. This ensures that the RDB file is newly generated.

   For example, the default RDB policy configurations in the **redis.conf** file are as follows:

   ```
   save 900 1 //Writes changed data into an RDB file if there is any data change within 900s.
   save 300 10 //Writes changed data into an RDB file if there are more than 10 data changes within 300s.
   save 60 10000 //Writes changed data into an RDB file if there are more than 10,000 data changes within 60s.
   ```

   Based on the preceding policy configurations, after stopping your service systems from writing data into the Redis instances, you can write test data to trigger the policies, so that all service data can be synchronized to the RDB file.

   You can delete the test data after data import.

   NOTE

   - If there is any DB not used by your service systems, you can write test data into the DB, and run the **flushdb** command to clear the database after importing data into DCS.
   - Compared with master/standby instances, single-node instances without data persistence configured require a longer time for export of an RDB file, because the RDB file is temporarily generated.

2. Log in to the ECS.
3. Install redis-cli. The following steps assume that your client is installed on the Linux OS.

   a. Run the following command to download Redis: You can also install other Redis versions. For details, see the **Redis official website**.

      ```
      wget http://download.redis.io/releases/redis-5.0.8.tar.gz
      ```

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

      ```
      tar -xzf redis-5.0.8.tar.gz
      ```

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

      ```
      cd redis-5.0.8
      make
      cd src
      ```

4. Run the following command to export the RDB file:

   ```
   redis-cli -h {source_redis_address} -p {port} -a {password} --rdb {output.rdb}
   ```

   {source_redis_address} is the connection address of the source Redis, *{port}* is the port of the source Redis, *{password}* is the connection password of the source Redis, and *{output.rdb}* is the RDB file name.

   If "Transfer finished with success." is displayed after the command is executed, the file is exported successfully.

Uploading the RDB File to Huawei Cloud ECS

To save time, you are advised to compress the RDB file and upload it to Huawei Cloud ECS using an appropriate mode (for example, SFTP mode).

NOTE

Ensure that the ECS has sufficient disk space for data file decompression, and can communicate with the DCS instance. Generally, the ECS and DCS instance are configured to belong to the same VPC and subnet, and the configured security group rules do not restrict access ports.

Importing Data

Use redis-shake to import data.

It takes 4 to 10 seconds to import an RDB file of 1 million data (20 bytes per data segment) to a VPC.

Verifying the Migration

After the data is imported successfully, access the DCS instance and run the **info** command to check whether the data has been successfully imported as required. Connect to Redis by referring to **redis-cli**.

If the import fails, check the procedure. If the import command is incorrect, run the **flushall** or **flushdb** command to clear the cache data in the target instance, modify the import command, and try again.

10.4.5 Self-Hosted Redis Cluster Migration with redis-shake (Online)

redis-shake is an open-source tool for migrating data online or offline (by importing backup files) between Redis Clusters. Data can be migrated to DCS Redis Cluster instances seamlessly because DCS Redis Cluster inherits the native Redis Cluster design.

The following describes how to use Linux redis-shake to migrate self-hosted Redis Cluster to a DCS Redis Cluster instance online.

Notes and Constraints

- To migrate data from a self-hosted Redis Cluster instance to a DCS Redis Cluster instance online, ensure that the source Redis is connected to the target Redis, or use a transit cloud server to connect the source and target cluster instances.

Prerequisites

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.
- An Elastic Cloud Server (ECS) has been created. For details about how to create an ECS, see **Purchasing an ECS**.

  Select the same VPC, subnet, and security group as the DCS Redis Cluster instance, and bind EIPs to the ECS.
- If the source self-hosted Redis Cluster is deployed on cloud servers of another cloud, allow public access to the servers.

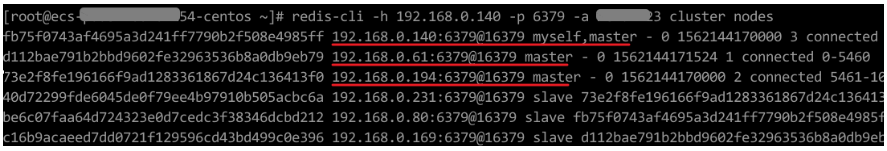
Obtaining Information of the Source and Target Redis Nodes

1. Connect to the source and target Redis instances, respectively. Connect to Redis by referring to **redis-cli**.
2. In online migration of Redis Clusters, the migration must be performed node by node. Run the following command to query the IP addresses and ports of all nodes in both the source and target Redis Clusters.

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

   {redis_address} indicates the Redis connection address, *{redis_port}* indicates the Redis port, and *{redis_password}* indicates the Redis connection password.

   In the command output similar to the following, obtain the IP addresses and ports of all masters.

   ```
   [root@ecs-[redacted]54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a [redacted]23 cluster nodes
   fb75f0743af4695a3d241ff7790b2f508e4985ff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
   d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
   73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-10
   40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
   be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
   c16b9acaeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
   ```

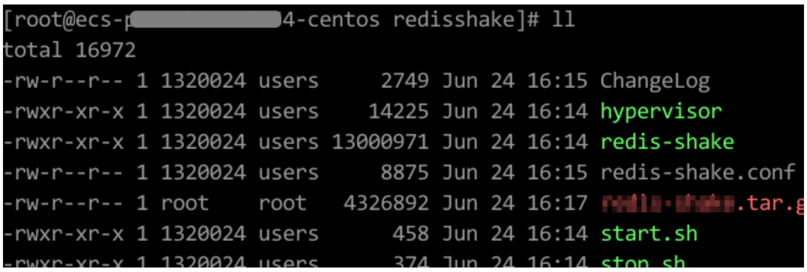
Configuring the redis-shake Tool

1. Log in to the ECS.
2. Run the following command on the ECS to download the redis-shake: This section uses v2.1.2 as an example. You can also download **other redis-shake versions** as required.

   ```
   wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.1.2-20220329/
   release-v2.1.2-20220329.tar.gz
   ```

3. Decompress the redis-shake file.

```
tar -xvf redis-shake-v2.1.2.tar.gz
```



```
[root@ecs-[redacted]4-centos redisshake]# ll
total 16972
-rw-r--r-- 1 1320024 users     2749 Jun 24 16:15 ChangeLog
-rwxr-xr-x 1 1320024 users    14225 Jun 24 16:14 hypervisor
-rwxr-xr-x 1 1320024 users 13000971 Jun 24 16:14 redis-shake
-rw-r--r-- 1 1320024 users     8875 Jun 24 16:15 redis-shake.conf
-rw-r--r-- 1 root    root   4326892 Jun 24 16:17 [redacted].tar.g
-rwxr-xr-x 1 1320024 users      458 Jun 24 16:14 start.sh
-rwxr-xr-x 1 1320024 users      374 Jun 24 16:14 stop.sh
```

4. Go to the redis-shake directory.

```
cd redis-shake-v2.0.3
```

5. Edit the **redis-shake.conf** file by providing the following information about all the masters of both the source and the target:

```
vim redis-shake.conf
```

The modification is as follows:

```
source.type = cluster
# If there is no password, skip the following parameter.
source.password_raw = {source_redis_password}
# IP addresses and port numbers of all masters of the source Redis Cluster, which are separated by semicolons (;).
source.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:{masterN_port}
target.type = cluster
# If there is no password, skip the following parameter.
target.password_raw = {target_redis_password}
# IP addresses and port numbers of all masters of the target instance, which are separated by semicolons (;).
target.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:{masterN_port}
```

Press **Esc** to exit the editing mode and enter **:wq!**. Press **Enter** to save the configuration and exit the editing interface.

Migrating Data Online

Run the following command to synchronize data between the source and the target Redis:

```
./redis-shake -type sync -conf redis-shake.conf
```

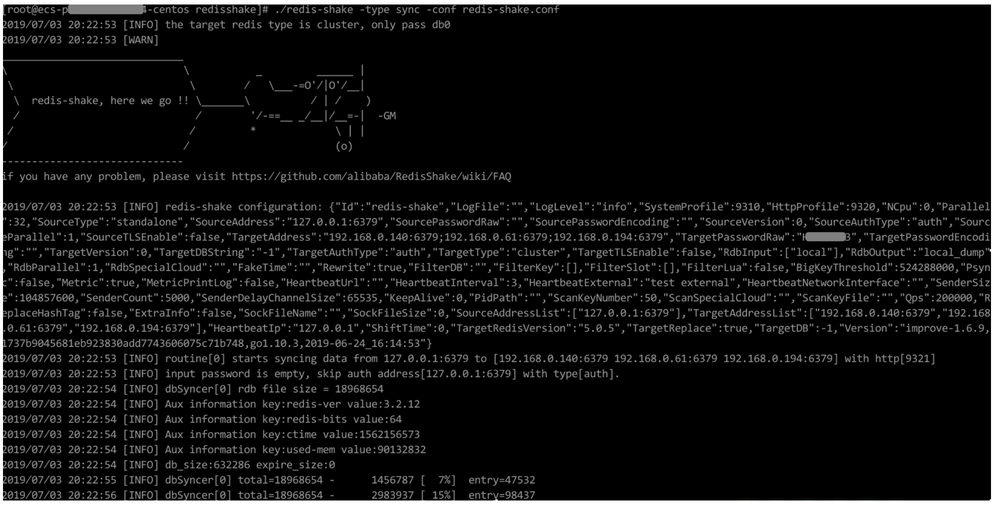
If the following information is displayed, the full synchronization has been completed and incremental synchronization begins.

```
sync rdb done.
```

If the following information is displayed, no new data is incremented. You can stop the incremental synchronization by pressing **Ctrl**+**C**.

```
sync:  +forwardCommands=0  +filterCommands=0  +writeBytes=0
```

Figure 10-8 Online migration using redis-shake



```
[root@ecs-p[redacted]-centos redisshake]# ./redis-shake -type sync -conf redis-shake.conf
2019/07/03 20:22:53 [INFO] the target redis type is cluster, only pass db0
2019/07/03 20:22:53 [WARN]
______________________________
\                             \           _         ______ |
 \                             \        /   \___-=O'/|O'/__|
  \  redis-shake, here we go !! \_______\          / | /    )
  /                             /        '/-==__ _/__|/__=-|  -GM
 /                             /         *             \ | |
/                             /                        (o)
------------------------------
if you have any problem, please visit https://github.com/alibaba/RedisShake/wiki/FAQ

2019/07/03 20:22:53 [INFO] redis-shake configuration: {"Id":"redis-shake","LogFile":"","LogLevel":"info","SystemProfile":9310,"HttpProfile":9320,"NCpu":0,"Parallel
":32,"SourceType":"standalone","SourceAddress":"127.0.0.1:6379","SourcePasswordRaw":"","SourcePasswordEncoding":"","SourceVersion":0,"SourceAuthType":"auth","Sourc
eParallel":1,"SourceTLSEnable":false,"TargetAddress":"192.168.0.140:6379;192.168.0.61:6379;192.168.0.194:6379","TargetPasswordRaw":"[redacted]","TargetPasswordEncodi
ng":"","TargetVersion":0,"TargetDBString":"-1","TargetAuthType":"auth","TargetType":"cluster","TargetTLSEnable":false,"RdbInput":["local"],"RdbOutput":"local_dump"
,"RdbParallel":1,"RdbSpecialCloud":"","FakeTime":"","Rewrite":true,"FilterDB":"","FilterKey":[],"FilterSlot":[],"FilterLua":false,"BigKeyThreshold":524288000,"Psyn
c":false,"Metric":true,"MetricPrintLog":false,"HeartbeatUrl":"","HeartbeatInterval":3,"HeartbeatExternal":"test external","HeartbeatNetworkInterface":"","SenderSiz
e":104857600,"SenderCount":5000,"SenderDelayChannelSize":65535,"KeepAlive":0,"PidPath":"","ScanKeyNumber":50,"ScanSpecialCloud":"","ScanKeyFile":"","Qps":200000,"R
eplaceHashTag":false,"ExtraInfo":false,"SockFileName":"","SockFileSize":0,"SourceAddressList":["127.0.0.1:6379"],"TargetAddressList":["192.168.0.140:6379","192.168
.0.61:6379","192.168.0.194:6379"],"HeartbeatIp":"127.0.0.1","ShiftTime":0,"TargetRedisVersion":"5.0.5","TargetReplace":true,"TargetDB":-1,"Version":"improve-1.6.9,
1737b9045681eb923830add7743606075c71b748,go1.10.3,2019-06-24_16:14:53"}
2019/07/03 20:22:53 [INFO] routine[0] starts syncing data from 127.0.0.1:6379 to [192.168.0.140:6379 192.168.0.61:6379 192.168.0.194:6379] with http[9321]
2019/07/03 20:22:53 [INFO] input password is empty, skip auth address[127.0.0.1:6379] with type[auth].
2019/07/03 20:22:54 [INFO] dbSyncer[0] rdb file size = 18968654
2019/07/03 20:22:54 [INFO] Aux information key:redis-ver value:3.2.12
2019/07/03 20:22:54 [INFO] Aux information key:redis-bits value:64
2019/07/03 20:22:54 [INFO] Aux information key:ctime value:1562156573
2019/07/03 20:22:54 [INFO] Aux information key:used-mem value:90132832
2019/07/03 20:22:54 [INFO] db_size:632286 expire_size:0
2019/07/03 20:22:55 [INFO] dbSyncer[0] total=18968654 -      1456787 [  7%]  entry=47532
2019/07/03 20:22:56 [INFO] dbSyncer[0] total=18968654 -      2983937 [ 15%]  entry=98437
```

Verifying the Migration

1. After the data synchronization, connect to the Redis Cluster DCS instance by referring to **redis-cli**.
2. Run the **info** command to check whether the data has been successfully imported as required.

   If the data has not been fully imported, run the **flushall** or **flushdb** command to clear the cached data in the target instance, and migrate data again.
3. After the verification is complete, delete the redis shake configuration file.

10.4.6 Self-Hosted Redis Cluster Migration with redis-shake (RDB)

redis-shake is an open-source tool for migrating data online or offline (by importing backup files) between Redis Clusters. Data can be migrated to DCS Redis Cluster instances seamlessly because DCS Redis Cluster inherits the native Redis Cluster design.

The following describes how to use Linux redis-shake to migrate self-hosted Redis Cluster to a DCS Redis Cluster instance offline.

📖 NOTE

If the source Redis and the target Redis cannot be connected, or the source Redis is deployed on other clouds, you can migrate data by importing backup files.

Prerequisites

- A DCS Redis Cluster instance has been created. For details about how to create one, see **Buying a DCS Redis Instance**.

  The memory of the target Redis instance cannot be smaller than that of the source Redis.
- An Elastic Cloud Server (ECS) has been created. For details about how to create an ECS, see **Purchasing an ECS**. Select the same VPC, subnet, and security group as the DCS Redis Cluster instance.

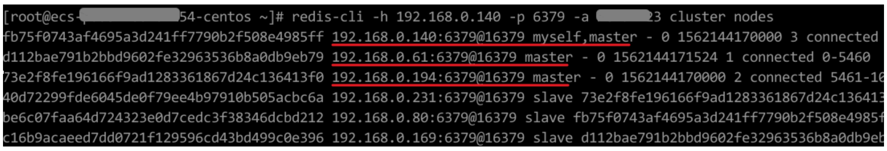
Obtaining Information of the Source and Target Redis Nodes

1. Connect to the source and target Redis instances, respectively. Connect to Redis by referring to **redis-cli**.
2. In online migration of Redis Clusters, the migration must be performed node by node. Run the following command to query the IP addresses and ports of all nodes in both the source and target Redis Clusters.

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

   {redis_address} indicates the Redis connection address, *{redis_port}* indicates the Redis port, and *{redis_password}* indicates the Redis connection password.

   In the command output similar to the following, obtain the IP addresses and ports of all masters.

   ```
   [root@ecs-[redacted]54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a [redacted]23 cluster nodes
   fb75f0743af4695a3d241ff7790b2f508e4985ff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
   d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
   73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-10
   40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
   be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
   c16b9acaeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
   ```

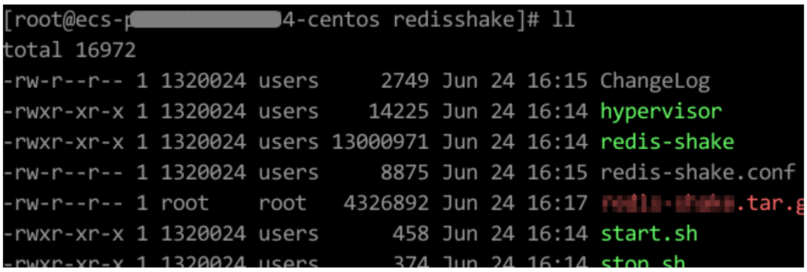
Installing RedisShake

1. Log in to the ECS.
2. Run the following command on the ECS to download the redis-shake: This section uses v2.1.2 as an example. You can also download **other redis-shake versions** as required.

   ```
   wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.1.2-20220329/
   release-v2.1.2-20220329.tar.gz
   ```

3. Decompress the redis-shake file.

   ```
   tar -xvf redis-shake-v2.1.2.tar.gz
   ```

   ```
   [root@ecs-[redacted]4-centos redisshake]# ll
   total 16972
   -rw-r--r-- 1 1320024 users     2749 Jun 24 16:15 ChangeLog
   -rwxr-xr-x 1 1320024 users    14225 Jun 24 16:14 hypervisor
   -rwxr-xr-x 1 1320024 users 13000971 Jun 24 16:14 redis-shake
   -rw-r--r-- 1 1320024 users     8875 Jun 24 16:15 redis-shake.conf
   -rw-r--r-- 1 root    root   4326892 Jun 24 16:17 [redacted].tar.g
   -rwxr-xr-x 1 1320024 users      458 Jun 24 16:14 start.sh
   -rwxr-xr-x 1 1320024 users      374 Jun 24 16:14 stop.sh
   ```

NOTE

If the source cluster is deployed in the data center intranet, install redis-shake on the intranet server. Export the source cluster backup file by referring to **Exporting the Backup File**. Upload the backup to the cloud server as instructed by the following steps

Exporting the Backup File

1. Go to the redis-shake directory.

   ```
   cd redis-shake-v2.0.3
   ```

2. Edit the **redis-shake.conf** file by providing the following information about all the masters of the source:

   ```
   vim redis-shake.conf
   ```

   The modification is as follows:

   ```
   source.type = cluster
   # If there is no password, skip the following parameter.
   source.password_raw = {source_redis_password}
   # IP addresses and port numbers of all masters of the source Redis Cluster, which are separated by
   ```

```
semicolons (;).
source.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:
{masterN_port}
```

Press **Esc** to exit the editing mode and enter **:wq!**. Press **Enter** to save the configuration and exit the editing interface.

3. Run the following command to export the RDB file:

```
./redis-shake -type dump -conf redis-shake.conf
```

If the following information is displayed in the execution log, the backup file is exported successfully:

```
execute runner[*run.CmdDump] finished!
```

Importing the Backup File

1. Import the RDB file (or files) to the cloud server. The cloud server must be connected to the target DCS instance.
2. Edit the **redis-shake.conf** file by providing the following information about all the masters of the target:

```
vim redis-shake.conf
```

The modification is as follows:

```
target.type = cluster
# If there is no password, skip the following parameter.
target.password_raw = {target_redis_password}
# IP addresses and port numbers of all masters of the target instance, which are separated by
semicolons (;).
target.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:
{masterN_port}
# List the RDB files to be imported, separated by semicolons (;).
rdb.input = {local_dump.0};{local_dump.1};{local_dump.2};{local_dump.3}
```

Press **Esc** to exit the editing mode and enter **:wq!**. Press **Enter** to save the configuration and exit the editing interface.

3. Run the following command to import the RDB file to the target instance:

```
./redis-shake -type restore -conf redis-shake.conf
```

If the following information is displayed in the execution log, the backup file is imported successfully:

```
Enabled http stats, set status (incr), and wait forever.
```

Verifying the Migration

1. After the data synchronization, connect to the Redis Cluster DCS instance by referring to **redis-cli**.
2. Run the **info** command to check whether the data has been successfully imported as required.

   If the data has not been fully imported, run the **flushall** or **flushdb** command to clear the cached data in the target instance, and migrate data again.
3. After the verification is complete, delete the redis shake configuration file.

10.5 Migration from Another Cloud

10.5.1 Migrating Redis from Another Cloud Online

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 another cloud to the target DCS.

NOTE

During online migration, data is essentially synchronized in full to a new replica. Therefore, perform online migration during low-demand hours. Otherwise, source instance CPU usage may surge and latency may increase.

Notes and Constraints

- If the **SYNC** and **PSYNC** commands are disabled by the source instance, enable them by contacting the source vendor. Otherwise, the migration fails.
- 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.
- The source must be Redis 3.0 or later.
- For earlier instances whose passwords contain single quotation marks ('), modify the password for online migration or try other methods.

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 multi-DB single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to **Enabling Multi-DB**.
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to **Online Migration with Rump**.
- The IP address and port of the source Redis instance has been obtained.
- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Creating an Online Migration Task

Step 1 Log in to the DCS console using the account of the target DCS Redis instance.

Step 2 Click 📍 in the upper left corner of the console and select the region where your target 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.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

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

NOTE

- **Select the same VPC as the target Redis. Ensure that the migration resource can access the target Redis instance.**
- The online 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

Checking 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 **Configuring the 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 **Configuring the Online Migration Task**. If no, go to **Step 3**.

Step 3 If the source and target Redis instances are on different clouds, create a connection using only Direct Connect. For details, see **Direct Connect documentation**.

----End

Configuring the Online Migration Task

Step 1 On the **Online Migration** tab page, click **Configure** in the row containing the online migration task you just created.

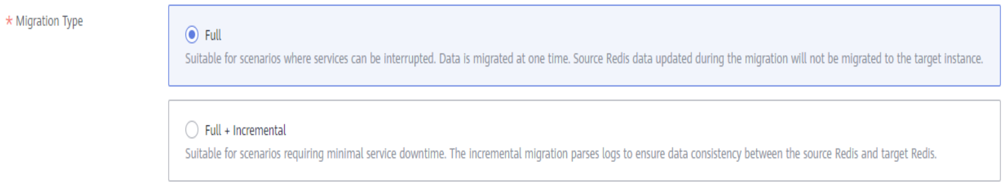
Step 2 Select a migration type.

Supported migration types are **Full** and **Full + Incremental**, which are described in **Table 10-7**.

Table 10-7 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 10-9 Selecting the migration type



* Migration Type

● Full

Suitable for scenarios where services can be interrupted. Data is migrated at one time. Source Redis 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 Redis and target Redis.

Step 3 Only 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**.

1. Configure **Source Redis Type** and **Source Redis Instance**:

   Set **Redis in the cloud** for **Source Redis Type** and add **Source Redis Instance**.

   If the source Redis is a Redis Cluster, enter the IP addresses and ports of all masters in the cluster and separate multiple addresses with commas (,). For example: **192.168.1.1:6379,192.168.0.0:6379**

2. Configure **Target Redis Type** and **Target Redis Instance**:

   Set **Redis in the cloud** for **Target Redis Type** and add **Target Redis Instance**.

3. Configure **Source Redis Instance Password** and **Target Redis Instance Password**: 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. If the test fails, check whether the password is correct, and whether the migration task network is connected.

   If a DCS Redis instance is used, the users created in **Managing Users** are currently unavailable.

4. You can specify the source DB and target DB. For example, if you enter **5** for source DB and **6** for target DB, data in DB5 of the source Redis will be migrated to DB6 of the target Redis. If the source DB is not specified but the target DB is specified, all source data will be migrated to the specified target DB by default. If the target DB is not specified, data will be migrated to the corresponding target DB.

   NOTE

   – If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.

   – For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

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**.

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

NOTE

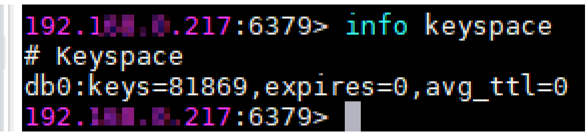
- Once incremental migration starts, it remains **Migrating** after full migration.
- To manually stop a migration task, select the check box on the left of the migration task and click **Stop** above the migration task.

----End

Verifying the Migration

Before data migration, if the target Redis has no data, check data integrity after the migration is complete in the following way:

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



```
192.1[redacted].[redacted].217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.[redacted].[redacted].217:6379>
```

3. Calculate the differences 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.

10.5.2 Backup Import from Another Cloud

This section describes how to migrate Redis from another cloud to DCS by importing backup files.

Simply download the source Redis data and then upload the data to an OBS bucket in the same account and 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.

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 multi-DB single-node or master/standby instance to a Proxy Cluster instance, check whether any data exists on databases other than DB0. If yes, enable multi-DB for the Proxy Cluster instance by referring to **Enabling Multi-DB**.
- By default, a Redis Cluster instance has only one DB (DB0). Before you migrate data from a multi-DB single-node or master/standby instance to a Redis Cluster instance, check whether any data exists on databases other than DB0. To ensure that the migration succeeds, move all data to DB0 by referring to **Online Migration with Rump**.
- Prepare the source Redis backup file. The backup file must be in .aof, .rdb, .zip, or .tar.gz format.
- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.

Creating an OBS Bucket and Uploading Backup Files

If the source Redis backup file to be uploaded is smaller than 5 GB, perform the following steps to create an OBS bucket and upload the file on the OBS console. If the backup file to be uploaded is larger than 5 GB, upload the file by referring to **instructions**.

Step 1 Create an OBS bucket on the OBS console.

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*.

1. **Region**:

   The OBS bucket must be in the same region as the target DCS Redis instance.

2. **Storage Class**: Available options are **Standard**, **Infrequent Access**, and **Archive**.

   Do not select **Archive**. Otherwise, the migration will fail.

Step 2 In the bucket list, click the bucket created in **Step 1**.

Step 3 In the navigation pane, choose **Objects**.

Step 4 On the **Objects** tab page, click **Upload Object**.

Step 5 Specify **Storage Class**.

Do not select **Archive**. Otherwise, the migration will fail.

Step 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.

Step 7 (Optional) Select **KMS encryption** to encrypt the uploaded files.

Step 8 Click **Upload**.

----End

Creating a Migration Task

Step 1 Go to the DCS console.

Step 2 In the navigation pane, choose **Data Migration**.

Step 3 Click **Create Backup Import Task**.

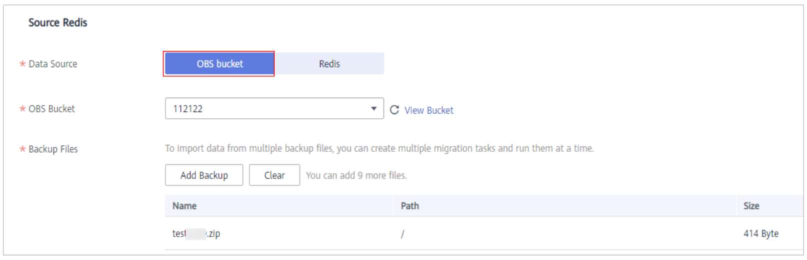
Step 4 Enter the task name and description.

The task name must start with a letter, contain 4 to 64 characters, and contain only letters, digits, hyphens (-), and underscores (_).

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.

Figure 10-10 Specifying the backup file information



Source Redis

* Data Source: OBS bucket | Redis

* OBS Bucket: 112122 ▾ View Bucket

* Backup Files: To import data from multiple backup files, you can create multiple migration tasks and run them at a time.

Add Backup Clear You can add 9 more files.

Name	Path	Size
tes[redacted].zip	/	414 Byte

Step 7 In the **Target Redis** area, select the **Target Redis Instance** prepared in **Prerequisites**.

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 For **Target DB**, you can specify a DB in the target Redis to migrate data to. For example, if you enter **5**, data will be migrated to DB5 of the target Redis. If you do not specify a DB, data will be migrated to a DB corresponding to the source DB.

NOTE

- If the source Redis is multi-DB and the target is single-DB (DB0), either ensure that all source data is in DB0, or specify a source DB and set the target DB to **0**. Otherwise, migration will fail.
- For details about DB in DCS for Redis, see **Does DCS for Redis Support Multi-DB?**

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

10.5.3 Online Migration from Another Cloud Using Rump

Redis instances provided by some cloud service vendors do not allow **SLAVEOF**, **BGSAVE**, and **PSYNC** commands to be issued from Redis clients. As a result, redis-cli, redis-shake, and other tools cannot be used to export data. Using the **KEYS** command may block Redis. Cloud service vendors usually only support downloading backup files. This method is suitable only for offline migration, featuring longer service interruption.

Rump is an open-source tool designed for migrating Redis data online. It supports migration between DBs of the same instance and between DBs of different instances. This section describes how to migrate another cloud to DCS by using Rump.

Migration Principles

Rump uses the **SCAN** command to acquire keys and the **DUMP**/**RESTORE** command to get or set values.

Featuring time complexity O(1), **SCAN** is capable of quickly getting all keys. **DUMP**/**RESTORE** is used to read/write values independent from the key type.

Rump brings the following benefits:

- The **SCAN** command replaces the **KEYS** command to avoid blocking Redis.
- Any type of data can be migrated.
- **SCAN** and **DUMP**/**RESTORE** operations are pipelined, improving the network efficiency during data migration.
- No temporary file is involved, saving disk space.

- Buffered channels are used to optimize performance of the source server.

Notes and Constraints

- The target cannot be a cluster DCS instance.
- To prevent migration command resolution errors, do not include special characters (#@:) in the instance password.
- Before data migration, suspend your services. If data is kept being written in during the migration, some keys might be lost.

Prerequisites

- 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. For comparing migration data and reserving sufficient memory, you are advised to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**. If any data exists on the target instance, duplicate data between the source and target is overwritten. If the data exists only on the target instance, the data will be retained.
- An Elastic Cloud Server (ECS) has been created. For details about how to create an ECS, see **Purchasing an ECS**.

  Select the same VPC, subnet, and security group as the DCS Redis Cluster instance, and bind EIPs to the ECS.

Installing the Rump

1. Log in to the ECS.
2. Download **Rump (release version)**.

   On 64-bit Linux, run the following command:

   ```
   wget https://github.com/stickermule/rump/releases/download/0.0.3/rump-0.0.3-linux-amd64;
   ```
3. After decompression, run the following commands to add the execution permission:

   ```
   mv rump-0.0.3-linux-amd64 rump;
   chmod +x rump;
   ```

Migrating Data

Run the following command to migrate data:

```
rump -from {source_redis_address}  -to {target_redis_address}
```

- *{source_redis_address}*

  Source Redis instance address, in the format of redis://[user:password@]host:port/db. **[user:password@]** is optional. If the instance is accessed in password-protected mode, you must specify the password in the RFC 3986 format. **user** can be omitted, but the colon (:) cannot be omitted. For example, the address may be **redis://:mypassword@192.168.0.45:6379/1**.

  db is the sequence number of the database. If it is not specified, the default value is 0.
- {*target_redis_address*}

  Address of the target Redis instance, in the same format as the source.

In the following example, data in DB0 of the source Redis is migrated to the target Redis whose connection address is 192.168.0.153. ****** stands for the password.

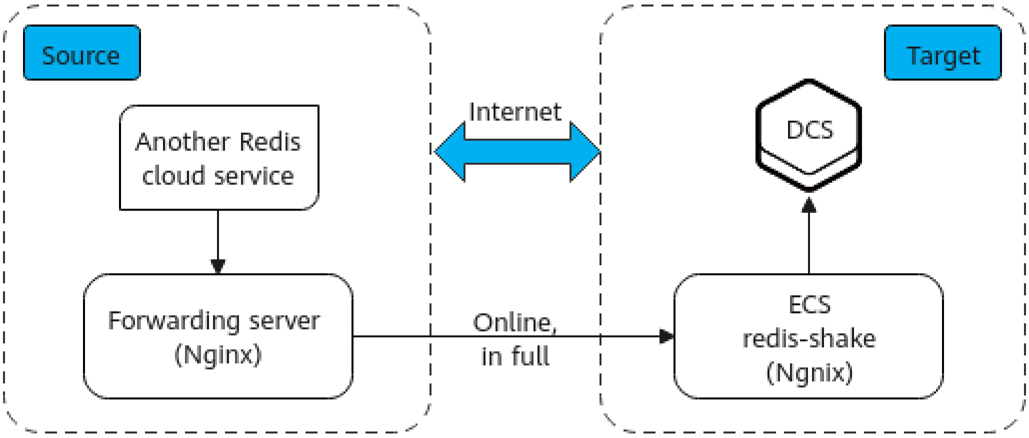
```
[root@ecs ~]# ./rump -from redis://127.0.0.1:6379/0  -to redis://:******@192.168.0.153:6379/0
.Sync done.
[root@ecs ~]#
```

10.5.4 Migrating from Another Cloud Online Using redis-shake

redis-shake is an open-source Redis migration tool. Its **rump** mode allows you to obtain the full data of a source Redis using the **SCAN** command and write the data to a target Redis. This migration solution does not involve the **SYNC** or **PSYNC** command and can be widely used for migration between self-built Redis and cloud Redis.

This section describes how to use the **rump** mode of redis-shake to migrate the full Redis data of another cloud service vendor at a time online to Huawei Cloud DCS.

Figure 10-11 Data flow in this solution



Notes and Constraints

- The **rump** mode does not support incremental data migration. To keep data consistency, stop writing data to the source Redis before migration.
- This solution applies only to same-database mapping and does not apply to inter-database mapping.
- If the source Redis has multiple databases (there are databases other than DB0), and your Huawei Cloud DCS instance is Proxy Cluster, multi-DB must be enabled for the DCS instance. Otherwise, the migration will fail. (Single-DB Proxy Cluster instances do not support the **SELECT** command.)
- If the source Redis has multiple databases (there are databases other than DB0), and your Huawei Cloud DCS instance is Redis Cluster, this solution cannot be used. (Redis Cluster DCS instances support only DB0.)

Prerequisites

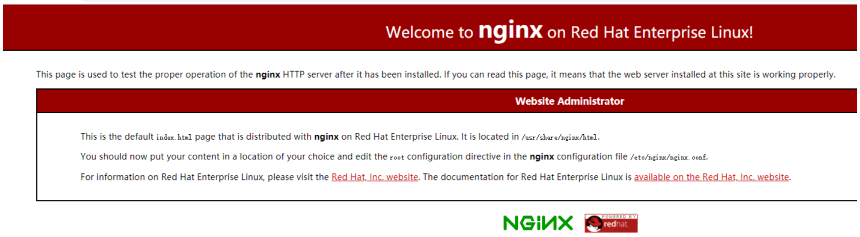
- A **DCS Redis instance** has been created.
- An **ECS** has been created for running redis-shake. The ECS must use the same VPC as the Redis instance, and be bound to EIPs.

Procedure

Step 1 Install Nginx on the Huawei Cloud ECS and the source forwarding server. The following describes how to install Nginx on an ECS running CentOS 7.x. The commands vary depending on the OS.

1. Add Nginx to the Yum repository.
```
sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm
```
2. Check whether Nginx has been added successfully.
```
yum search nginx
```
3. Install Nginx.
```
sudo yum install -y nginx
```
4. Install the stream module.
```
yum install nginx-mod-stream --skip-broken
```
5. Start Nginx and set it to run automatically upon system startup.
```
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
```
6. In the address box of a browser, enter the server address (the EIP of the ECS) to check whether Nginx is installed successfully.

   If the following page is displayed, Nginx has been installed successfully.



Welcome to **nginx** on Red Hat Enterprise Linux!

This page is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

Website Administrator

This is the default `index.html` page that is distributed with **nginx** on Red Hat Enterprise Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

For information on Red Hat Enterprise Linux, please visit the Red Hat, Inc. website. The documentation for Red Hat Enterprise Linux is available on the Red Hat, Inc. website.

Step 2 Add the source forwarding server to the whitelist of the source Redis.

Step 3 Configure a security group for the source forwarding server.

1. Obtain the EIP of the Huawei Cloud ECS.
2. In the inbound rule of the security group of the source forwarding server, add the EIP of the Huawei Cloud ECS, and open the port that Huawei Cloud ECS's requests come through. The following takes port 6379 as an example.

Step 4 Configure Nginx forwarding for the source forwarding server.

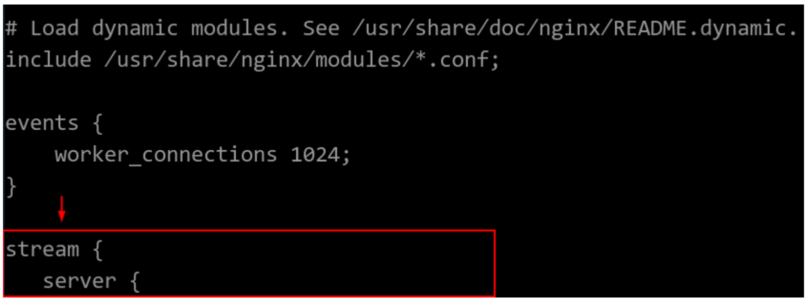
1. Log in to the Linux source forwarding server and run the following commands to open and modify the configuration file:
```
cd /etc/nginx
vi nginx.conf
```
2. Example forwarding configuration:
```
stream {
    server {
        listen 6379;
        proxy_pass {source_instance_address}:{port};
    }
  }
```

6379 is the listening port of the source forwarding server. *{source_instance_address}* and *{port}* are the connection address and port of the source Redis instance.

This configuration allows you to access the source Redis through the local listening port 6379 of the source forwarding server.

This configuration must be added exactly where it is shown in the following figure.

Figure 10-12 Configuration location



```
# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

stream {
   server {
```

3. Restart Nginx.

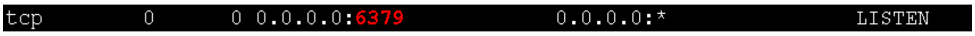
```
service nginx restart
```

4. Verify whether Nginx has been started.

```
netstat -an|grep 6379
```

If the port is being listened, Nginx has been started successfully.

Figure 10-13 Verification result



```
tcp        0      0 0.0.0.0:6379            0.0.0.0:*               LISTEN
```

Step 5 Configure Nginx forwarding for the Huawei Cloud ECS.

1. Log in to the Linux ECS on Huawei Cloud and run the following commands to open and modify the configuration file:

```
cd /etc/nginx
vi nginx.conf
```

2. Configuration example:

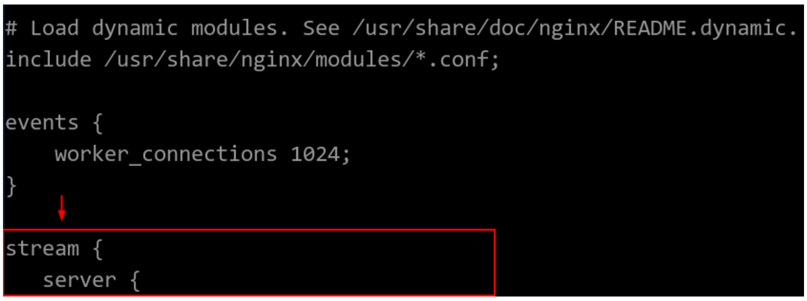
```
stream {
    server {
        listen 6666;
        proxy_pass {source_ecs_address}:6379;
    }
  }
```

6666 is Huawei Cloud ECS's listening port, *{source_ecs_address}* is the public IP address of the source forwarding server, and **6379** is the listening port of the source forwarding server Nginx.

This configuration allows you to access the source forwarding server through the local listening port 6666 of the Huawei Cloud ECS.

This configuration must be added exactly where it is shown in the following figure.

Figure 10-14 Configuration location



```
# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

stream {
   server {
```

3. Restart Nginx.

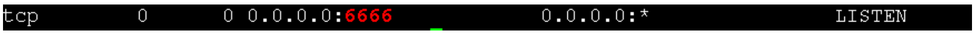
```
service nginx restart
```

4. Verify whether Nginx has been started.

```
netstat -an|grep 6666
```

   If the port is being listened, Nginx has been started successfully.

Figure 10-15 Verification result



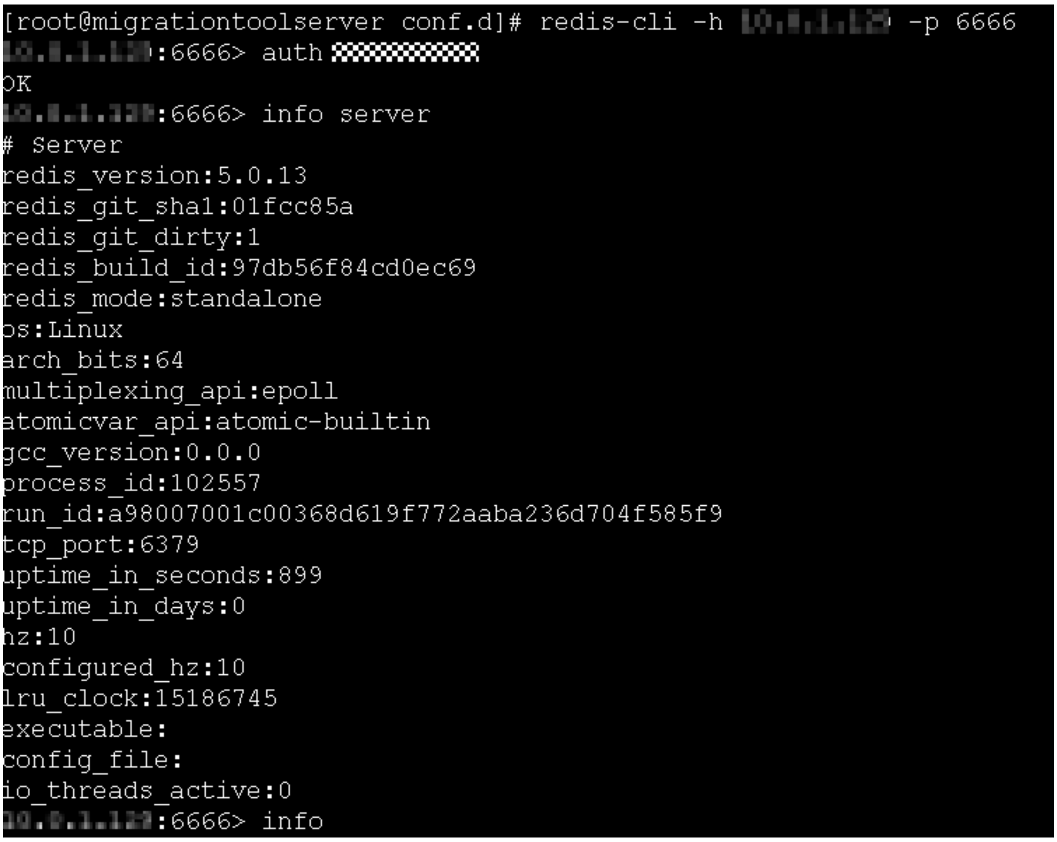
```
tcp        0      0 0.0.0.0:6666            0.0.0.0:*               LISTEN
```

Step 6 Run the following command on the Huawei Cloud ECS to test the network connection of port 6666:

```
redis-cli -h {target_ecs_address} -p 6666 -a {password}
```

{target_ecs_address} is the EIP of the Huawei Cloud ECS, **6666** is the listening port of the Huawei Cloud ECS, and *{password}* is the source Redis password. If there is no password, leave it blank.

Figure 10-16 Connection example



```
[root@migrationtoolserver conf.d]# redis-cli -h [redacted] -p 6666
[redacted]:6666> auth [redacted]
OK
[redacted]:6666> info server
# Server
redis_version:5.0.13
redis_git_sha1:01fcc85a
redis_git_dirty:1
redis_build_id:97db56f84cd0ec69
redis_mode:standalone
os:Linux
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:0.0.0
process_id:102557
run_id:a98007001c00368d619f772aaba236d704f585f9
tcp_port:6379
uptime_in_seconds:899
uptime_in_days:0
hz:10
configured_hz:10
lru_clock:15186745
executable:
config_file:
io_threads_active:0
[redacted]:6666> info
```

Step 7 Prepare the migration tool redis-shake.

1. Log in to the Huawei Cloud ECS.
2. Download redis-shake on the Huawei Cloud ECS. Version 2.0.3 is used as an example. You can use **other redis-shake versions** as required.
   ```
   wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.0.3-20200724/
   redis-shake-v2.0.3.tar.gz
   ```
3. Decompress the redis-shake file.
   ```
   tar -xvf redis-shake-v2.0.3.tar.gz
   ```

Step 8 Configure the redis-shake configuration file.

1. Go to the directory generated after the decompression.
   ```
   cd redis-shake-v2.0.3
   ```
2. Modify the **redis-shake.conf** configuration file.
   ```
   vim redis-shake.conf
   ```
   Modify the source Redis configuration.
   - source.type

     Type of the source Redis instance. Use **standalone** for single-node, master/standby, and Proxy Cluster, and **cluster** for cluster instances.
   - source.address

     EIP of the Huawei Cloud ECS and the mapped port of the source forwarding server (Huawei Cloud ECS's listening port 6666). Separate the EIP and port number with a colon (:).
   - source.password_raw

     Password of the source Redis instance. If no password is set, you do not need to set this parameter.

   Modify the target DCS configuration.
   - target.type

     Type of the DCS Redis instance. Use **standalone** for single-node, master/standby, and Proxy Cluster, and **cluster** for cluster instances.
   - target.address

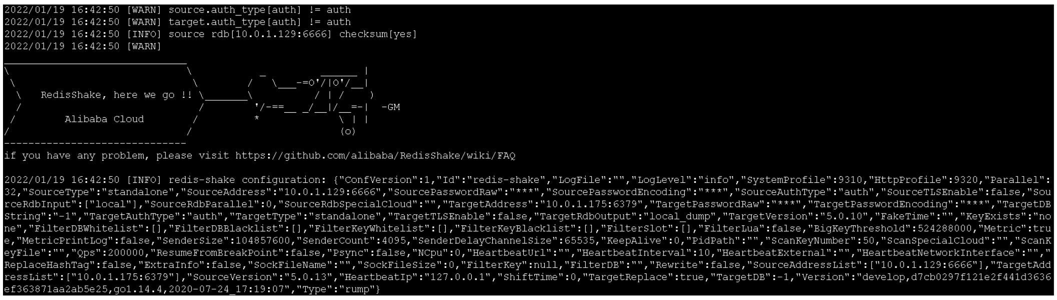
     Colon (:) separated connection address and port of the DCS Redis instance.
   - target.password_raw

     Password of the DCS Redis instance. If no password is set, you do not need to set this parameter.
3. Press **Esc** to exit the editing mode and enter **:wq!**. Press **Enter** to save the configuration and exit the editing interface.

Step 9 Run the following command to start redis-shake and migrate data in the **rump** (online in full) mode:

```
./redis-shake.linux -conf redis-shake.conf -type rump
```

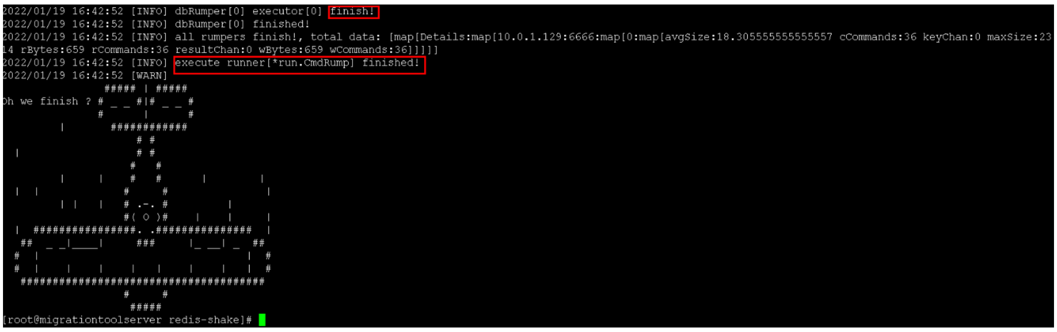
Figure 10-17 Migration process



```
2022/01/19 16:42:50 [WARN] source.auth_type[auth] != auth
2022/01/19 16:42:50 [WARN] target.auth_type[auth] != auth
2022/01/19 16:42:50 [INFO] source rdb[10.0.1.129:6666] checksum[yes]
2022/01/19 16:42:50 [WARN]
______________________________
\                             \           _         ______ |
 \                             \        /   \___-=O'/|O'/__|
  \   RedisShake, here we go !! \_______\          / | /    )
  /                             /        '/-==__ _/__|/__=-|  -GM
 /        Alibaba Cloud        /         *             \ | |
/                             /                        (o)
------------------------------
if you have any problem, please visit https://github.com/alibaba/RedisShake/wiki/FAQ

2022/01/19 16:42:50 [INFO] redis-shake configuration: {"ConfVersion":1,"Id":"redis-shake","LogFile":"","LogLevel":"info","SystemProfile":9310,"HttpProfile":9320,"Parallel":
32,"SourceType":"standalone","SourceAddress":"10.0.1.129:6666","SourcePasswordRaw":"***","SourcePasswordEncoding":"***","SourceAuthType":"auth","SourceTLSEnable":false,"Sou
rceRdbInput":["local"],"SourceRdbParallel":0,"SourceRdbSpecialCloud":"","TargetAddress":"10.0.1.175:6379","TargetPasswordRaw":"***","TargetPasswordEncoding":"***","TargetDB
String":"-1","TargetAuthType":"auth","TargetType":"standalone","TargetTLSEnable":false,"TargetRdbOutput":"local_dump","TargetVersion":"5.0.10","FakeTime":"","KeyExists":"no
ne","FilterDBWhitelist":[],"FilterDBBlacklist":[],"FilterKeyWhitelist":[],"FilterKeyBlacklist":[],"FilterSlot":[],"FilterLua":false,"BigKeyThreshold":524288000,"Metric":tru
e,"MetricPrintLog":false,"SenderSize":104857600,"SenderCount":4095,"SenderDelayChannelSize":65535,"KeepAlive":0,"PidPath":"","ScanKeyNumber":50,"ScanSpecialCloud":"","ScanK
eyFile":"","Qps":200000,"ResumeFromBreakPoint":false,"Psync":false,"NCpu":0,"HeartbeatUrl":"","HeartbeatInterval":10,"HeartbeatExternal":"","HeartbeatNetworkInterface":"","
ReplaceHashTag":false,"ExtraInfo":false,"SockFileName":"","SockFileSize":0,"FilterKey":null,"FilterDB":"","Rewrite":false,"SourceAddressList":["10.0.1.129:6666"],"TargetAdd
ressList":["10.0.1.175:6379"],"SourceVersion":"5.0.13","HeartbeatIp":"127.0.0.1","ShiftTime":0,"TargetReplace":true,"TargetDB":-1,"Version":"develop,d7cb0297f121e2f441d3636
ef363871aa2ab5e25,go1.14.4,2020-07-24_17:19:07","Type":"rump"}
```

Figure 10-18 Migration result



```
2022/01/19 16:42:52 [INFO] dbRumper[0] executor[0] finish!
2022/01/19 16:42:52 [INFO] dbRumper[0] finished!
2022/01/19 16:42:52 [INFO] all rumpers finish!, total data: [map[Details:map[10.0.1.129:6666:map[0:map[avgSize:18.305555555555557 cCommands:36 keyChan:0 maxSize:23
14 rBytes:659 rCommands:36 resultChan:0 wBytes:659 wCommands:36]]]]]
2022/01/19 16:42:52 [INFO] execute runner[*run.CmdRump] finished!
2022/01/19 16:42:52 [WARN]
                ##### | #####
Oh we finish ? # _ _ #|# _ _ #
               #      |      #
         |       ############
                     # #
  |                  # #
                    #   #
         |     |    #   #      |        |
  |  |             #     #               |
         | |   |   # .-. #          |
                   #( O )#    |     |     |
  |  ################. .###############  |
   ##  _ _|_____|     ###     |_ __| _  ##
  #  |                                |  #
  #  |    |    |    |   |    |    |   |  #
   ######################################
                   #     #
                    #####
[root@migrationtoolserver redis-shake]#
```

Step 10 After the migration is complete, use redis-cli to connect to the source and target Redis instances to check whether the data is complete.

1. Connect to the source and target Redis instances, respectively.

   For details, see **Access Using redis-cli**.

2. Run the **info keyspace** command to check the values of **keys** and **expires**.
3. Calculate the differences 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.

Step 11 Delete the redis-shake configuration file.

----End

10.5.5 Backup Import from Another Cloud Using redis-shake

redis-shake is an open-source tool for migrating data online or offline (by importing backup files) between Redis Clusters. If the source Redis Cluster is deployed in another cloud, and online migration is not supported, you can migrate data by importing backup files.

If the source Redis and the target Redis cannot be connected, or the source Redis is deployed on other clouds, you can migrate data by importing backup files.

The following describes how to use redis-shake for backup migration to a DCS Redis Cluster instance.

Prerequisites

- A **DCS Redis instance** has been created. Note that the memory of the DCS Redis Cluster instance cannot be smaller than that of the source cluster.

- An **ECS** has been created for running redis-shake. The ECS must use the same VPC, subnet, and security group as the Redis instance.

Procedure

1. Access the target Redis instance using **redis-cli**. Obtain the IP address and port of the master node of the target instance.

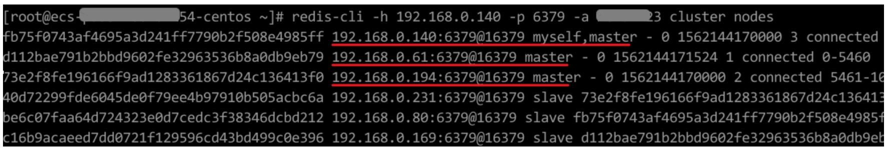
   ```
   redis-cli -h {target_redis_address} -p {target_redis_port} -a {target_redis_password} cluster nodes
   ```

   – *{target_redis_address}*: connection address of the target DCS Redis instance.

   – *{target_redis_port}*: port of the target DCS Redis instance.

   – *{target_redis_password}*: password for connecting to the target DCS Redis instance.

   In the command output similar to the following, obtain the IP addresses and ports of all masters.

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

2. Install redis-shake on the prepared Huawei Cloud ECS.

   a. Log in to the Huawei Cloud ECS.

   b. Download redis-shake on the Huawei Cloud ECS. Version 2.0.3 is used as an example. You can use **other redis-shake versions** as required.

   ```
   wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.0.3-20200724/redis-shake-v2.0.3.tar.gz
   ```

   c. Decompress the redis-shake file.

   ```
   tar -xvf redis-shake-v2.0.3.tar.gz
   ```

   NOTE

   If the source Redis is deployed in the data center intranet, install redis-shake on the intranet server. Export data and then upload the data to the cloud server as instructed by the following steps.

3. Export the RDB file from the source Redis console. If the RDB file cannot be exported, contact customer service of the source.

4. Import the RDB file.

   a. Import the RDB file (or files) to the cloud server. The cloud server must be connected to the target DCS instance.

   b. Edit the redis-shake configuration file **redis-shake.conf**.

   ```
   vim redis-shake.conf
   ```

   Add the following information about all the masters of the target:

   ```
   target.type = cluster
   # If there is no password, skip the following parameter.
   target.password_raw = {target_redis_password}
   # IP addresses and port numbers of all masters of the target instance, which are separated by semicolons (;).
   target.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}…{masterN_ip}:{masterN_port}
   # List the RDB files to be imported, separated by semicolons (;).
   rdb.input = {local_dump.0};{local_dump.1};{local_dump.2};{local_dump.3}
   ```

   Press **Esc** to exit the editing mode and enter **:wq!**. Press **Enter** to save the configuration and exit the editing interface.

c. Run the following command to import the RDB file to the target instance:

```
./redis-shake -type restore -conf redis-shake.conf
```

If the following information is displayed in the execution log, the backup file is imported successfully:

```
Enabled http stats, set status (incr), and wait forever.
```

5. Verify the migration.

   After data synchronization, access the target Redis Cluster DCS instance using redis-cli. Run the **info** command to query the number of keys in the **Keyspace** section to confirm that data has been fully imported.

   If the data has not been fully imported, run the **flushall** or **flushdb** command to clear the cached data in the instance, and synchronize data again.

11 Testing Instance Performance

11.1 Testing Redis Performance Using memtier_benchmark

memtier_benchmark is a command-line tool developed by Redis Labs. It can generate traffic in various modes and supports Redis. This tool provides multiple options and reporting features that can be easily used through the CLI. For details, visit **https://github.com/RedisLabs/memtier_benchmark**.

This section describes how to use memtier-benchmark to test the performance of a DCS Redis instance when running command **SET** or **GET** in a high-concurrency scenario.

Test Procedure

Step 1 Create a DCS Redis instance.

Step 2 Create three ECSs and configure the same AZ, VPC, subnet, and security group for the ECSs and the instance.

NOTE

Only one ECS is required for testing on a single-node or master/standby instance.

Step 3 Install **memtier_benchmark** on each ECS. CentOS 8.0 is used as an example.

1. Preparations
   a. Install the tools required for compilation.

      yum install -y autoconf

      yum install -y automake

      yum install -y make

      yum install -y gcc-c++

      yum install -y git

   b. Enable the PowerTools repository.

      dnf config-manager --set-enabled PowerTools

c. Install the required dependencies.

yum install -y pcre-devel

yum install -y zlib-devel

yum install -y libmemcached-devel

yum install -y openssl-devel

2. Install the libevent library.

   yum install -y libevent-devel

3. Download, compile, and install the memtier_benchmark library.

   a. Create a folder in the root directory where the memtier_benchmark library will be stored.

      mkdir /env

   b. Download the memtier_benchmark source code.

      cd /env

      git clone https://github.com/RedisLabs/memtier_benchmark.git

   c. Go to the directory where the source code is located.

      cd memtier_benchmark

   d. Compile the source code and generate the executable file **memtier_benchmark**.

      autoreconf -ivf

      ./configure

      make

   e. Install the tool in the system.

      make install

4. Run the following command to check whether the installation is successful: If a parameter description of memtier_benchmark is returned, the installation is successful.

   memtier_benchmark --help

Step 4 Run the following test command on all ECSs:

```
memtier_benchmark -s {IP} -n {nreqs}  -c {connect_number}  -t 4 -d {datasize}
```

NOTE

Run **memtier_benchmark --cluster-mode -s *{IP}* -n *{nreqs}* -c *{connect_number}* -t 4 -d *{datasize}*** for a Redis Cluster instance.

Reference values: **-c *{connect_number}*: 200**; **-n *{nreqs}*: 10,000,000**; **-d *{datasize}*: 32**

- **-s** indicates the domain name or IP address of the instance.
- **-t** indicates the number of threads used in the benchmark test.
- **-c** indicates the number of client connections.
- **-d** indicates the size of a single data record in bytes.
- **-n** indicates the number of test packets.

Step 5 Repeat **Step 4** with different client connections to obtain the maximum QPS (Query per Second, number of read and write operations per second).

Step 6 The sum of operations per second of all the three ECSs indicates the performance of the instance specification.

To test on a Redis Cluster instance, launch two benchmark tools on each ECS.

----End

11.2 Testing Redis Performance Using redis-benchmark

The Redis client includes redis-benchmark, a performance testing utility that simulates N clients concurrently sending M number of query requests.

This section describes how to use redis-benchmark to test the performance of a DCS Redis instance when running command **SET** or **GET** in a high-concurrency scenario.

Test Procedure

Step 1 Create a DCS Redis instance.

Step 2 Create three ECSs and configure the same AZ, VPC, subnet, and security group for the ECSs and the instance.

NOTE

Only one ECS is required for testing on a single-node or master/standby instance.

Step 3 Install redis-benchmark on each ECS. The Redis server can be installed in either of the following ways and benchmark will be installed, too.

- Method 1:
  a. Download a Redis client. This example uses redis-6.0.9.

     wget http://download.redis.io/releases/redis-6.0.9.tar.gz

  b. Decompress the client installation package.

     tar xzf redis-6.0.9.tar.gz

  c. Go to the **src** directory of redis-6.0.9.

     cd redis-6.0.9/src

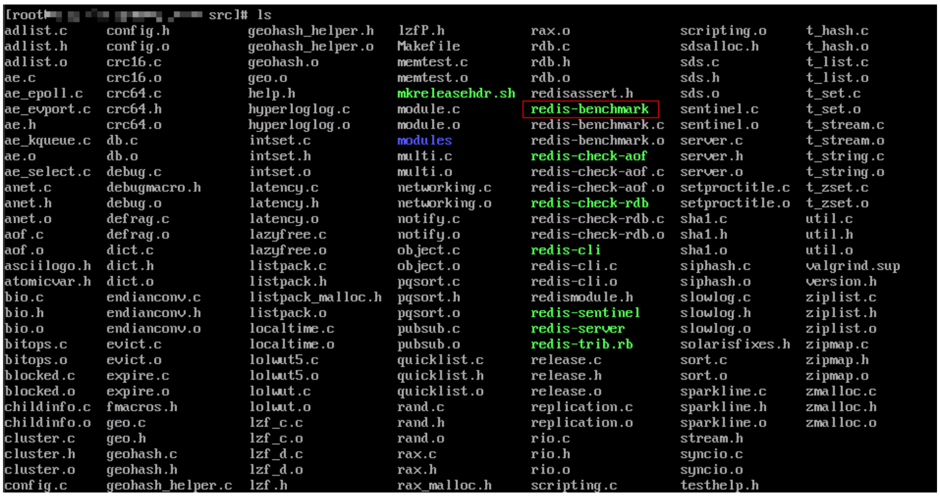
  d. Compile the source code.

     make

     After the compilation is complete, the tool is stored in the **src** directory of **redis-***x.x.x*.

  e. Check whether the redis-benchmark executable file exists.

     ls



```
[root[redacted] src]# ls
adlist.c     config.h          geohash_helper.h   lzfP.h           rax.o              scripting.o     t_hash.c
adlist.h     config.o          geohash_helper.o   Makefile         rdb.c              sdsalloc.h      t_hash.o
adlist.o     crc16.c           geohash.o          memtest.c        rdb.h              sds.c           t_list.c
ae.c         crc16.o           geo.o              memtest.o        rdb.o              sds.h           t_list.o
ae_epoll.c   crc64.c           help.h             mkreleasehdr.sh  redisassert.h      sds.o           t_set.c
ae_evport.c  crc64.h           hyperloglog.c      module.c         redis-benchmark    sentinel.c      t_set.o
ae.h         crc64.o           hyperloglog.o      module.o         redis-benchmark.c  sentinel.o      t_stream.c
ae_kqueue.c  db.c              intset.c           modules          redis-benchmark.o  server.c        t_stream.o
ae.o         db.o              intset.h           multi.c          redis-check-aof    server.h        t_string.c
ae_select.c  debug.c           intset.o           multi.o          redis-check-aof.c  server.o        t_string.o
anet.c       debugmacro.h      latency.c          networking.c     redis-check-aof.o  setproctitle.c  t_zset.c
anet.h       debug.o           latency.h          networking.o     redis-check-rdb    setproctitle.o  t_zset.o
anet.o       defrag.c          latency.o          notify.c         redis-check-rdb.c  sha1.c          util.c
aof.c        defrag.o          lazyfree.c         notify.o         redis-check-rdb.o  sha1.h          util.h
aof.o        dict.c            lazyfree.o         object.c         redis-cli          sha1.o          util.o
asciilogo.h  dict.h            listpack.c         object.o         redis-cli.c        siphash.c       valgrind.sup
atomicvar.h  dict.o            listpack.h         pqsort.c         redis-cli.o        siphash.o       version.h
bio.c        endianconv.c      listpack_malloc.h  pqsort.h         redismodule.h      slowlog.c       ziplist.c
bio.h        endianconv.h      listpack.o         pqsort.o         redis-sentinel     slowlog.h       ziplist.h
bio.o        endianconv.o      localtime.c        pubsub.c         redis-server       slowlog.o       ziplist.o
bitops.c     evict.c           localtime.o        pubsub.o         redis-trib.rb      solarisfixes.h  zipmap.c
bitops.o     evict.o           lolwut5.c          quicklist.c      release.c          sort.c          zipmap.h
blocked.c    expire.c          lolwut5.o          quicklist.h      release.h          sort.o          zipmap.o
blocked.o    expire.o          lolwut.c           quicklist.o      release.o          sparkline.c     zmalloc.c
childinfo.c  fmacros.h         lolwut.o           rand.c           replication.c      sparkline.h     zmalloc.h
childinfo.o  geo.c             lzf_c.c            rand.h           replication.o      sparkline.o     zmalloc.o
cluster.c    geo.h             lzf_c.o            rand.o           rio.c              stream.h
cluster.h    geohash.c         lzf_d.c            rax.c            rio.h              syncio.c
cluster.o    geohash.h         lzf_d.o            rax.h            rio.o              syncio.o
config.c     geohash_helper.c  lzf.h              rax_malloc.h     scripting.c        testhelp.h
```

f. Install the tool in the system.

   make install

- Method 2:

  Install the Redis server matching the ECS OS. The following examples use Ubuntu and CentOS.

  – Ubuntu

    ```
    sudo apt update
    sudo apt install redis-server
    ```

  – CentOS

    ```
    sudo yum install epel-release
    sudo yum update
    sudo yum -y install redis
    ```

Step 4 Run the following test command on all ECSs:

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connect_number} -d
{datasize} -t {command}
```

Reference values: **-c *{connect_number}***: **200**; **-n *{nreqs}***: **10,000,000**; **-r *{randomkeys}***: **1,000,000**; **-d *{datasize}***: **32**

- **-h**: instance domain name or IP address
- **-p**: port of the instance. The default value is **6379**.
- **-a**: password for connecting to the instance. This parameter is not required for password-free instances.
- **-t** Set of commands to be executed For example, to test only the **set** command, use **-t set**. To test the **ping**, **get**, and **set** commands, use **-t ping,set,get**. Use commas (,) to separate commands.
- **-c** number of client connections
- **-d** size of a single data record in bytes
- **-n** number of test packets
- **-r** number of random keys

Step 5 Repeat **Step 4** with different client connections to obtain the maximum QPS (Query per Second, number of read and write operations per second).

Step 6 The sum of operations per second of all the three ECSs indicates the performance of the instance specification.

To test on a Redis Cluster instance, launch two benchmark tools on each ECS.

NOTE

- Add the **--cluster** parameter only when testing Redis Cluster instances using redis-benchmark.
- In a test for the maximum number of connections of a Redis Cluster instance, if the performance of the ECSs is insufficient, the program will exit or the error message "Cannot assign requested address" will be displayed when the number of connections reaches 10,000. In this case, check how many ECSs are used in the test. Prepare three ECSs and start three redis-benchmark processes on each ECS.

----End

Common redis-cli Options

- **-h** ***<hostname>***: host name of the server, which can be an IP address or a domain name.
- **-p** ***<port>***: port of the server. The default value is **6379**.
- **-a** ***<password>***: password for connecting to the server. This parameter is not required for password-free instances.
- **-r** ***<repeat>***: number of times that a command is run.
- **-n** ***<db>***: DB number. The default value is **0**.
- **-c**: cluster mode (with **-ASK** and **-MOVED** redirections).
- **--latency**: a loop where latency is measured continuously.
- **--scan**: scans the key space without blocking the Redis server. (By contrast, scanning using **KEYS *** blocks Redis server).
- **--eval** ***<file>***: sends the **EVAL** command using a Lua script.
- **-x**: reads the last parameter in stdin.
- **--bigscan**: scans big keys in the data set.
- **--raw**: forces raw data output from the hexadecimal format, such as **\xe4\xb8**.

For more information about redis-cli, visit **https://redis.io/docs/connect/cli/**

Examples of Common redis-cli Commands

- Connect to an instance:

  ./redis-cli -h *{IP}* -p 6379
- Connect to a specified DB:

  ./redis-cli -h *{IP}* -p 6379 -n 10
- Connect to a Redis Cluster instance:

  ./redis-cli -h *{IP}* -p 6379 -c
- Test the latency (by sending the **ping** command):

  ./redis-cli -h *{IP}* -p 6379 --latency
- Scan for keys that match the specified pattern:

  ./redis-cli -h *{IP}* -p 6379 --scan --pattern '*:12345*'

Common Options in redis-benchmark (redis-6.0.9)

- **-h** ***<hostname>***: host name of the server, which can be an IP address or a domain name.
- **-p** ***<port>***: port of the server. The default value is **6379**.
- **-a** ***<password>***: password for connecting to the server. This parameter is not required for password-free instances.
- **-c** ***<clients>***: number of concurrent connections. The default value is **50**.
- **-n** ***<requests>***: total number of requests. The default value is **100000**.
- **-d** ***<size>***: data size of the **SET**/**GET** value, in bytes. The default value is **2**.
- **--dbnum** *<db >*: database number. The default value is **0**.
- **--threads** *<num>*: multi-thread mode, which is supported only by redis-benchmark compiled in Redis 6.0. In pressure tests, the multi-thread mode outperforms the single-thread mode.
- **--cluster**: cluster mode (required only by Redis Cluster).
- **-k** ***<boolean>***: **1**=keep alive; **0**=reconnect. The default value is **1**, indicating that both pconnect and connect can be tested.
- **-r** ***<keyspacelen>***: uses random keys for **SET**, **GET**, and **INCR**, and random values for **SADD**. *keyspacelen* indicates the number of keys to be added.
- **-e**: displays server errors to stdout.
- **-q**: displays only the number of queries per second.
- **-l**: runs tests in loops.
- **-t** ***<tests>***: tests specified commands.
- **-I** : idle mode. Open *N* idle connections and wait.
- **-P** ***<numreq>***: concurrent pipeline requests. The default value is **1**.

For more information about redis-benchmark, visit **https://redis.io/docs/latest/operate/oss_and_stack/management/optimization/benchmarks/**.

Examples of Common redis-benchmark Commands

- Test single-node, master/standby, read/write splitting, and Proxy Cluster instances:

  ./redis-benchmark -h *{IP address or domain name}* -p 6379 -a *{password}*--threads *{num}* -n *{ nreqs }* -r *{ randomkeys }* -c *{clients}* -d *{datasize}* -t *{command}*
- Test Redis Cluster instances:

  ./redis-benchmark -h *{IP address or domain name}* -p 6379 -a *{password}*--threads *{num}* -n *{ nreqs }* -r *{ randomkeys }* -c *{clients}* -d *{datasize}* --cluster -t *{command}*
- Test connect:

  ./redis-benchmark -h *{IP address or domain name}* -p 6379 -a *{password}*--threads *{num}* -n *{ nreqs }* -r *{ randomkeys }* -c *{clients}* -d *{datasize}* -k 0 -t *{command}*
- Test idle connections:

  ./redis-benchmark -h *{IP address or domain name}* -p 6379 -a *{pwd}* -c *{clients}* -I

11.3 Comparing redis-benchmark and memtier_benchmark

Tool	Memcached	Setting Read/Write Ratio	Random Payload	Setting Timeout
memtier_benchmark	Supported	Supported	Supported	Supported
redis-benchmark	Not supported	Not supported	Not supported	Not supported

11.4 Reference for a Redis Performance Test

11.4.1 Test Data of Master/Standby DCS Redis 4.0 or 5.0 Instances

Test Environment

- Redis instance specifications

  Redis 4.0 or 5.0 | 8 GB | master/standby

  Redis 4.0 or 5.0 | 32 GB | master/standby
- ECS flavors

  General computing-enhanced | c6.2xlarge.2 | 8 vCPUs | 16 GB
- ECS image

  Ubuntu 18.04 server 64-bit
- Test tool

  A single ECS is used for the test. The test tool is redis-benchmark.

Test Command

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t
{command}
```

Reference values: **-c *{connect_number}*: 500**; **-n *{nreqs}*: 10,000,000**; **-r *{randomkeys}*: 1,000,000**; **-d *{datasize}*: 32**; **-t *{command}*: set**

Test Result

NOTE

- The following test results are for reference only. The performance may vary depending on the site environment and network fluctuation.
- QPS: Query per second, indicates number of read and write operations per second. Unit: count/second.
- Average Latency: Average latency of operations, in milliseconds.
- xth Percentile Latency: latency of x% of operations, in milliseconds. For example, if the value is 10 ms, 99.99th percentile latency indicates that 99.99% queries can be processed within 10 ms.

Table 11-1 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
8 GB	x86	500	132,068.98	11	18	205	3.298
		10,000	82,386.58	171	178	263	69.275
32 GB	x86	500	131,385.33	9.5	16	17	3.333
		10,000	82,275.41	157	162.18	162.43	62.105

Table 11-2 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)	Average Latency (ms)
8 GB	x86	500	138,652.02	7	11	12	2.117
		10,000	82,710.94	123.7	281.6	282.9	61.078
32 GB	x86	500	139,113.02	6.6	10	11	2.119
		10,000	82,489.36	100	105.66	106	60.968

11.4.2 Test Data of Proxy Cluster DCS Redis 4.0 or 5.0 Instances

Test Environment

- Redis instance specifications

  Redis 4.0 or 5.0 | 64 GB | 8 shards | Proxy Cluster
- ECS flavors

  General computing-enhanced | c6.xlarge.2 | 4 vCPUs | 8 GB
- Test tool

  Three ECSs are used for concurrent tests. The test tool is memtier_benchmark.

Test Command

```
memtier_benchmark --ratio= (1:0 and 0:1) -s {IP} -n {nreqs}  -c {connect_number}  -t 4 -d {datasize}
```

Reference values: **-c *{connect_number}***: **1000**; **-n *{nreqs}***: **10,000,000**; **-d *{datasize}***: **32**

Test Result

NOTE

- The following test results are for reference only. The performance may vary depending on the site environment and network fluctuation.
- QPS: Query per second, indicates number of read and write operations per second. Unit: count/second.
- Average Latency: Average latency of operations, in milliseconds.
- xth Percentile Latency: latency of x% of operations, in milliseconds. For example, if the value is 10 ms, 99.99th percentile latency indicates that 99.99% queries can be processed within 10 ms.

Table 11-3 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	95th-Percentile Latency (ms)	99.99th-Percentile Latency (ms)	Maximum Latency (ms)
64 GB	x86	3000	1,323,935.00	3.3	9.4	220
		5000	1,373,756.00	5.3	13	240
		10,000	1,332,074.00	11	26	230
		80,000	946,032.00	110	460	6800

Table 11-4 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	95th-Percentile Latency (ms)	99.99th-Percentile Latency (ms)	Maximum Latency (ms)
64 GB	x86	3000	1,366,153.00	3.3	9.3	230
		5000	1,458,451.00	5.1	13	220
		10,000	1,376,399.00	11	29	440
		80,000	953,837.00	120	1300	2200

11.4.3 Test Data of Redis Cluster DCS Redis 4.0 or 5.0 Instances

Test Environment

- Redis instance specifications

  Redis 4.0 or 5.0 | 32 GB | Redis Cluster
- ECS flavors

  General computing-enhanced | c6.xlarge.2 | 4 vCPUs | 8 GB
- Test tool

  Three ECSs are used for concurrent tests. The test tool is memtier_benchmark.

Test Command

```
memtier_benchmark --cluster-mode --ratio=(1:0 and 0:1) -s {IP} -n {nreqs}  -c {connect_number}  -t 4 -d
{datasize}
```

Reference values: **-c *{connect_number}***: **1000**; **-n *{nreqs}***: **10,000,000**; **-d *{datasize}***: **32**

Test Result

NOTE

- The following test results are for reference only. The performance may vary depending on the site environment and network fluctuation.
- QPS: Query per second, indicates number of read and write operations per second. Unit: count/second.
- Average Latency: Average latency of operations, in milliseconds.
- xth Percentile Latency: latency of x% of operations, in milliseconds. For example, if the value is 10 ms, 99.99th percentile latency indicates that 99.99% queries can be processed within 10 ms.

Table 11-5 Test result of running the SET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
32 GB	x86	1000	371,780.2	5.6	6.3	44
		10,000	256,073.11	90	220	460
32 GB	Arm	1000	317,053.78	17	34	230
		10,000	248,832.33	410	490	750

Table 11-6 Test result of running the GET command

Redis Cache Size	CPU Type	Concurrent Connections	QPS	99.99th-Percentile Latency (ms)	First 100th-Percentile Latency (ms)	Last 100th-Percentile Latency (ms)
32 GB	x86	1000	427,000.04	5.0	5.3	78
		10,000	302,159.03	63	220	460
32 GB	Arm	1000	421,402.06	13	14	65
		10,000	309,359.18	180	260	500

11.4.4 Test Data of Master/Standby DCS Redis 6.0 Instances

DCS Redis 6.0 basic edition instances support SSL. This section covers the performance tested with and without SSL enabled.

Test Environment

- Redis instance specifications

  Redis 6.0 | Basic edition | 8 GB | Master/Standby

  Redis 6.0 | Basic edition | 32 GB | Master/Standby

- ECS flavors

  General compute-plus | 8 vCPUs | 16 GiB | c7.2xlarge.2

- ECS image

  Ubuntu 18.04 server 64-bit

- Test tool

  A single ECS is used for the test. The test tool is memtier_benchmark.

Test Command

SSL disabled:

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread}  -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize}
```

Reference values: **-c *{connect_number}***: **1000**, **--key-maximum*{max_key}***: **2000000**, **-d *{datasize}***: **32**

SSL enabled:

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread}  -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize}  --tls --cacert ca.crt
```

Reference values: **-c *{connect_number}***: **1000**, **--key-maximum*{max_key}***: **2000000**, **-d *{datasize}***: **32**

Test Result

NOTE

- The following test results are for reference only. The performance may vary depending on the site environment and network fluctuation.
- QPS: Query per second, indicates number of read and write operations per second. Unit: count/second.
- Average Latency: Average latency of operations, in milliseconds.
- xth Percentile Latency: latency of x% of operations, in milliseconds. For example, if the value is 10 ms, 99.99th percentile latency indicates that 99.99% queries can be processed within 10 ms.

Table 11-7 Test result of the SET command (SSL disabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
8 GB	x86	500	151,047.41	3.355	6.175	12.223
		1000	149,346.86	6.673	11.711	31.743
32 GB	x86	500	143,648.1	3.476	5.215	13.055
		4000	104,517.03	37.881	139.263	175.103

Table 11-8 Test result of the SET command (SSL enabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
8 GB	x86	500	86,827.84	5.537	8.575	9.535
		1000	92,413.99	10.055	15.615	17.279
32 GB	x86	500	87,385.5	5.584	8.383	9.343
		4000	50,813.67	62.623	100.863	104.959

Table 11-9 Test result of the GET command (SSL disabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
8 GB	x86	500	180,413.66	2.764	4.287	11.583
		1000	179,113.5	5.586	8.959	29.823
32 GB	x86	500	175,268.86	2.848	4.079	11.839
		4000	134,755.17	29.161	126.463	166.911

Table 11-10 Test result of the GET command (SSL enabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
8 GB	x86	500	113,637.22	4.316	6.239	7.359
		1000	105,504.55	8.962	13.439	15.295

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
32 GB	x86	500	100,309.99	4.603	6.559	6.943
		4000	57,007.69	55.052	85.503	89.087

11.4.5 Test Data of Redis Cluster DCS Redis 6.0 Instances

DCS Redis 6.0 basic edition instances support SSL. This section covers the performance tested with and without SSL enabled.

Test Environment

- Redis instance specifications

  Redis 6.0 | Basic edition | 32 GB | Redis Cluster
- ECS flavors

  General compute-plus | 8 vCPUs | 16 GiB | c7.2xlarge.2
- ECS image

  Ubuntu 18.04 server 64-bit
- Test tool

  Three ECSs are used for concurrent tests. The test tool is memtier_benchmark.

Test Command

SSL disabled:

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread}  -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize} --cluster-mode
```

Reference values: **-c *{connect_number}***: **1000**, **--key-maximum*{max_key}***: **2000000**, **-d *{datasize}***: **32**

SSL enabled:

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread}  -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize} --cluster-mode  --tls --cacert ca.crt
```

Reference values: **-c *{connect_number}***: **1000**, **--key-maximum*{max_key}***: **2000000**, **-d *{datasize}***: **32**

Test Result

NOTE

- The following test results are for reference only. The performance may vary depending on the site environment and network fluctuation.
- QPS: Query per second, indicates number of read and write operations per second. Unit: count/second.
- Average Latency: Average latency of operations, in milliseconds.
- xth Percentile Latency: latency of x% of operations, in milliseconds. For example, if the value is 10 ms, 99.99th percentile latency indicates that 99.99% queries can be processed within 10 ms.

Table 11-11 Test result of the SET command (SSL disabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
32 GB	x86	1000	322,899.21	2.661	4.319	8.511
		3000	360,336.14	7.757	13.055	29.439
		10,000	330,378.22	29.411	97.279	153,599

Table 11-12 Test result of the SET command (SSL enabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
32 GB	x86	1000	238,307.26	3.603	5.151	6.527
		3000	185,455.62	13.196	20.607	352.255
		10,000	111,913.19	57.537	96.767	121.343

Table 11-13 Test result of the GET command (SSL disabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
32 GB	x86	1000	450,422.66	1.875	2.767	6.879
		3000	432,450.2	6.451	12.095	28.415
		10,000	507,338.44	23.001	95.231	176.127

Table 11-14 Test result of the GET command (SSL enabled)

Redis Cache Size	CPU Type	Concurrent Connections	QPS	Average Latency (ms)	99th-Percentile Latency (ms)	99.9th-Percentile Latency (ms)
32 GB	x86	1000	274,066.16	3.076	4.255	7.071
		3000	201,063.51	11.743	18.047	387.071
		10,000	116,026.38	51.284	84.479	136.191

11.4.6 Test Data of Redis Backup, Restoration, and Migration

Test Environment

- Redis instance specifications
  Redis 5.0 | 8 GB | master/standby
  Redis 5.0 | 32 GB | master/standby
  Redis 5.0 | 64 GB | Proxy Cluster (2 replicas | 8 shards | 8 GB per shard)
  Redis 5.0 | 256 GB | Proxy Cluster (2 replicas | 32 shards | 8 GB per shard)
  Redis 5.0 | 64 GB | Redis Cluster (2 replicas | 8 shards | 8 GB per shard)
  Redis 5.0 | 256 GB | Redis Cluster (2 replicas | 32 shards | 8 GB per shard)
- ECS flavors
  c6s.large.2 2 vCPUs | 4 GB

Test Command

Run the following command on a 256 GB Proxy Cluster instance:

```
redis-benchmark - h {IP} -p{Port} -n 10000000 -r 10000000 -c 10000 -d 1024
```

Run the following command on a 256 GB Redis Cluster instance:

```
redis-benchmark - h {IP} -p{Port} -n 10000000 -r 10000000 -c 40000 -d 1024 -c
```

Test Result

Table 11-15 Migration

Source Instance Type	Source Instance Specifications (GB)	Target Instance Type	Target Instance Specifications (GB)	Migration Type	Data Volume (GB)	Duration (min)
Redis 5.0 | master/ standby	8	Redis 5.0 | master/ standby	8	Full + incremental	7.78	3
Redis 5.0 | master/ standby	32	Redis 5.0 | master/ standby	32	Full + incremental	31.9	17
Redis 5.0 | Proxy Cluster	64	Redis 5.0 | Proxy Cluster	64	Full + incremental	62.42	7
Redis 5.0 | Redis Cluster	64	Redis 5.0 | Redis Cluster	64	Full + incremental	57.69	6
Redis 5.0 | Proxy Cluster	256	Redis 5.0 | Proxy Cluster	256	Full + incremental	241.48	23
Redis 5.0 | Redis Cluster	256	Redis 5.0 | Redis Cluster	256	Full + incremental	240.21	22

Table 11-16 Backup

Instance Type	Instance Specifications (GB)	Backup Mode	Data Volume (GB)	Duration (min)
Redis 5.0 | master/ standby	8	RDB	7.78	2
Redis 5.0 | master/ standby	32	RDB	31.9	5

Instance Type	Instance Specifications (GB)	Backup Mode	Data Volume (GB)	Duration (min)
Redis 5.0 | Proxy Cluster	64	RDB	62.42	9
Redis 5.0 | Proxy Cluster	256	RDB	241.48	37
Redis 5.0 | Redis Cluster	64	RDB	57.69	9
Redis 5.0 | Redis Cluster	256	RDB	255	39
Redis 5.0 | master/standby	8	AOF	7.9	2
Redis 5.0 | master/standby	32	AOF	31.15	10
Redis 5.0 | Proxy Cluster	64	AOF	62.42	20
Redis 5.0 | Proxy Cluster	256	AOF	241.48	48
Redis 5.0 | Redis Cluster	64	AOF	57.69	19
Redis 5.0 | Redis Cluster	256	AOF	255	51

Table 11-17 Restoration

Instance Type	Instance Specifications (GB)	Restoration Mode	Data Volume (GB)	Duration (min)
Redis 5.0 | master/standby	8	RDB	7.9	2
Redis 5.0 | master/standby	32	RDB	31.15	6
Redis 5.0 | Proxy Cluster	64	RDB	62.42	10

Instance Type	Instance Specifications (GB)	Restoration Mode	Data Volume (GB)	Duration (min)
Redis 5.0 | Proxy Cluster	256	RDB	246	42
Redis 5.0 | Redis Cluster	64	RDB	57.69	10
Redis 5.0 | Redis Cluster	256	RDB	255	40
Redis 5.0 | master/standby	8	AOF	7.9	3
Redis 5.0 | master/standby	32	AOF	31.15	10
Redis 5.0 | Proxy Cluster	64	AOF	62.42	10
Redis 5.0 | Proxy Cluster	256	AOF	246	46
Redis 5.0 | Redis Cluster	64	AOF	57.69	10
Redis 5.0 | Redis Cluster	256	AOF	255	43

12 Applying for More DCS 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.

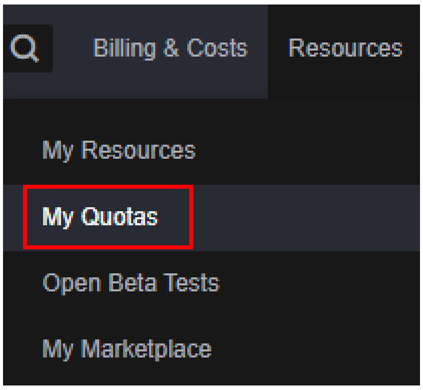
If a quota cannot meet your needs, apply for a higher quota.

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 12-1 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.

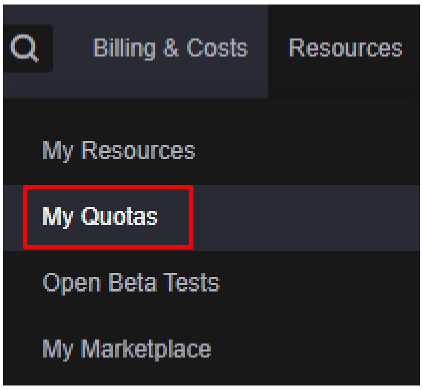
How Do I Increase My Quota?

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 12-2 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**.

13 Viewing Monitoring Metrics and Configuring Alarms

Cloud Eye is a secure, scalable monitoring platform. It monitors DCS metrics, and sends notifications if alarms are triggered or events occur.

13.1 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 13-1 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 13-2**.

Table 13-2 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. Includes connections from replicas and established for system monitoring, configuration synchronization, and services	≥ 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.	● **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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
expires	Keys With an Expiration	Number of keys with an expiration in Redis	≥ 0	DCS Redis instance	1 minute
keyspace_hits_perc	Hit Rate	Ratio of the number of Redis cache hits to the number of lookups. Calculation: keyspace_hits/(keyspace_hits + keyspace_misses) Aggregated from the master and replica nodes. If no read command is performed within a monitoring period, the ratio is 0. 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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
blocked_clients	Blocked Clients	Number of clients suspended by block operations	≥ 0	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	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	Current memory fragmentation	≥ 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 13-3 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_out_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_frag_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. Includes connections from replicas and established for system monitoring, configuration synchronization, and services	≥ 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	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

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	Redis Server of a master/standby, read/write splitting or cluster DCS Redis 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	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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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: keyspace_hits/ (keyspace_hits + keyspace_misses) If no read command is performed within a monitoring period, the ratio is 0. 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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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.	- **1**: yes - **0**: no	Redis Server of a master/standby, read/write splitting or cluster DCS Redis instance	1 minute
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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

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	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
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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 13-4 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.	● **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
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
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

Metric ID	Metric Name	Metric Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
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
dcs_cluster_redis_node	Redis Server
dcs_cluster_proxy2_node	Proxy in a Proxy Cluster and read/write splitting DCS Redis 4.0 or later instance

13.2 Common DCS Metrics

This section describes common Redis metrics.

Table 13-5 Common metrics

Metric	Description
CPU Usage	This metric indicates the maximum value in each measurement period (minute-level: every minute; second-level: every 5 seconds). ● 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	This metric measures the memory usage in each measurement period (minute-level: every minute; second-level: every 5 seconds). ● 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. **NOTICE** The memory usage does not include the usage of reserved memory.
Connected Clients	This metric indicates the number of instantaneous connected clients, that is, the number of concurrent connections. This metric does not include the number of connections to the standby nodes of master/standby or cluster instances. For details about the maximum allowed number of connections, see the "Max. Connections" column of different instance types listed in **DCS Instance Specifications**.
Ops per Second	This metric indicates the number of operations processed per second. 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**.
Input Flow	This metric indicates the instantaneous input traffic. ● 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.

Metric	Description
Output Flow	This metric indicates the instantaneous output traffic. ● 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	This metric indicates the percentage of the used bandwidth to the maximum bandwidth limit. Bandwidth usage = (Input flow + Output flow)/(2 x Maximum bandwidth) x 100%
Commands Processed	This metric indicates the number of commands processed during the monitoring period. The default monitoring period is 1 minute. 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.
Flow Control Times	This metric indicates the number of times that the maximum allowed bandwidth is exceeded during the monitoring period. For details about the maximum allowed bandwidth, see the "Maximum/Assured Bandwidth" column of different instance types listed in **DCS Instance Specifications**.
Slow Queries	This metric indicates whether slow queries exist on the instance. For details about the cause of a slow query, see **Viewing Redis Slow Queries**.

13.3 Viewing DCS Metrics

The Cloud Eye service monitors the running performance your DCS instances.

Viewing DCS Metrics

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

13.4 Configuring DCS Monitoring and Alarms

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 13-6 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.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
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.
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.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
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 DCS Redis Instances

Table 13-7 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.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
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, read/write splitting, or master/standby instance cannot be expanded. If you need larger capacity, use a cluster instance instead.
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).

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
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.
Bandwidth Usage	0–200%	Alarm threshold: > 90% Number of consecutive periods: 2 Alarm severity: Major	Yes	Check whether the bandwidth usage increase comes from read services or write services based on the input and output flow. If the bandwidth usage of a single node is high, check whether big keys exist. 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. 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.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
Flow Control Times	≥ 0	Alarm threshold: > 0 Number of consecutive periods: 1 Alarm severity: Critical	Yes	Consider capacity expansion based on the specification limits, input flow, and output flow.

Alarm Policies for Proxy Nodes of DCS Redis Instances

Table 13-8 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.

Metric	Value Range	Alarm Policy	Approach Upper Limit	Handling Suggestion
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

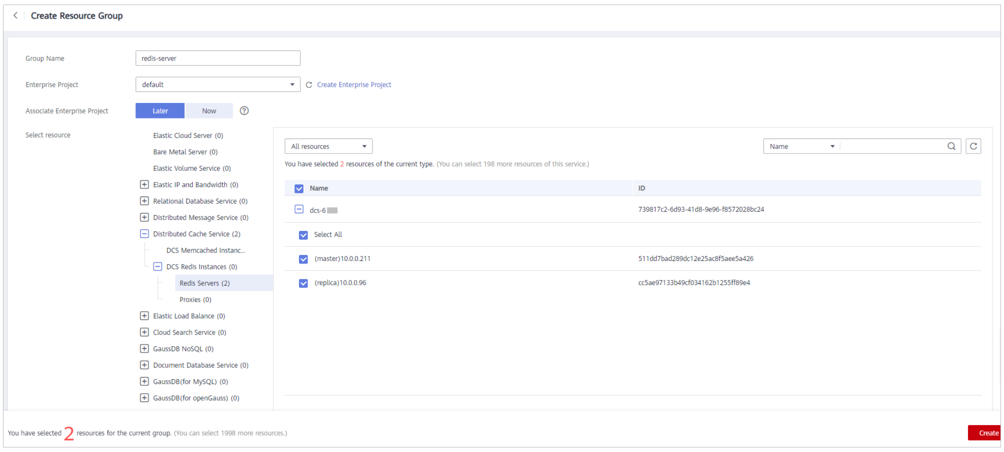
Cloud Eye allows you to add DCS instances, Redis Server nodes, and proxy nodes to resource groups and manage instances and alarm rules by group to simplify O&M. For details, see **Creating a Resource Group**.

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 13-1 Creating a resource group



Group Name: redis-server

Enterprise Project: default

Associate Enterprise Project: Later

Select resource

All resources

You have selected 2 resources of the current type. (You can select 198 more resources of this service.)

☑ Name	ID
dcs-6[redacted]	739817c2-6d93-41d8-9e96-f8572028bc24
☑ Select All	
☑ (master)10.0.0.211	511dd7bad289dc12e25ac8f5aee5a426
☑ (replica)10.0.0.96	cc5ae97133b49cf034162b1255ff89e4

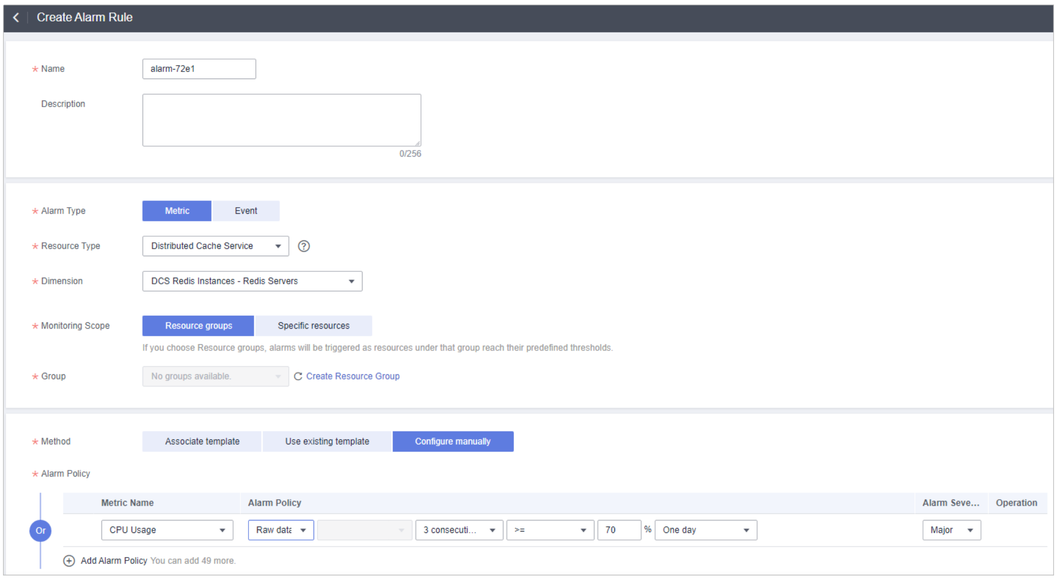
You have selected 2 resources for the current group. (You can select 1998 more resources.)

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 13-2 Creating an alarm rule for a resource group



Create Alarm Rule

* Name: alarm-72e1

Description: 0/256

* Alarm Type: Metric | Event

* Resource Type: Distributed Cache Service

* Dimension: DCS Redis Instances - Redis Servers

* Monitoring Scope: Resource groups | Specific resources

If you choose Resource groups, alarms will be triggered as resources under that group reach their predefined thresholds.

* Group: No groups available. Create Resource Group

* Method: Associate template | Use existing template | Configure manually

* Alarm Policy

Metric Name	Alarm Policy						Alarm Seve...	Operation
CPU Usage	Raw date		3 consecuti...	>=	70 %	One day	Major	

Or

Add Alarm Policy You can add 49 more.

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.

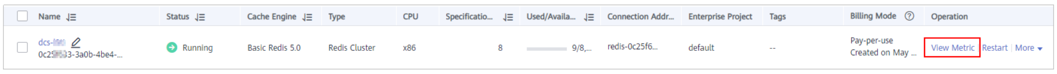
Step 1 Log in to the management console, and choose **Application** > **Distributed Cache Service** in the service list.

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 13-3 Viewing instance metrics



☐	Name	Status	Cache Engine	Type	CPU	Specificatio...	Used/Availa...	Connection Addr...	Enterprise Project	Tags	Billing Mode	Operation
☐	dcs-[redacted] 0c25[redacted]3-3a0b-4be4-...	Running	Basic Redis 5.0	Redis Cluster	x86	8	9/8,...	redis-0c25f6...	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.

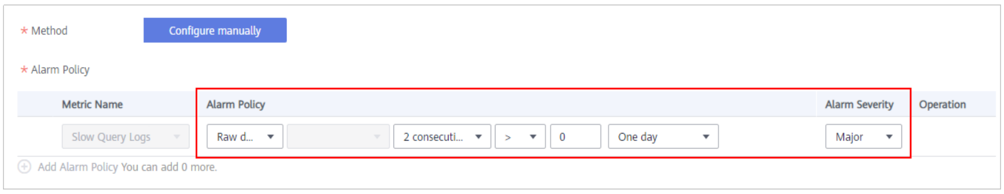
The **Create Alarm Rule** page is displayed.

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 13-4** 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 13-4 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

14 Viewing DCS Audit Logs

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.

DCS Operations Supported by CTS

Table 14-1 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
Changing instance password	Redis	modifyDCSInstancePassword

Operation	Resource Type	Trace Name
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	Instance	submitBatchStopDCSInstanceRequest
Restarting instances in batches	Redis	batchRestartDCSInstance
Submitting a request to restart instances in batches	Redis	submitBatchRestartDCSInstanceRequest
Starting multiple instances	Redis	batchStartDCSInstance

Operation	Resource Type	Trace Name
Submitting a request to start instances in batches	Instance	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
Creating an instance subscription order	Redis	createInstanceOrder

Operation	Resource Type	Trace Name
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
Changing the billing mode	Redis	billingModeChange
Updating instance tags	Redis	updateInstanceTag

Operation	Resource Type	Trace Name
Modifying the whitelist configuration	Instance	modifyWhiteList
Modifying instance bandwidth	Redis	modifyBandwidth

Viewing Audit Logs

View CTS logs of DCS, see **Querying Real-Time Traces**.