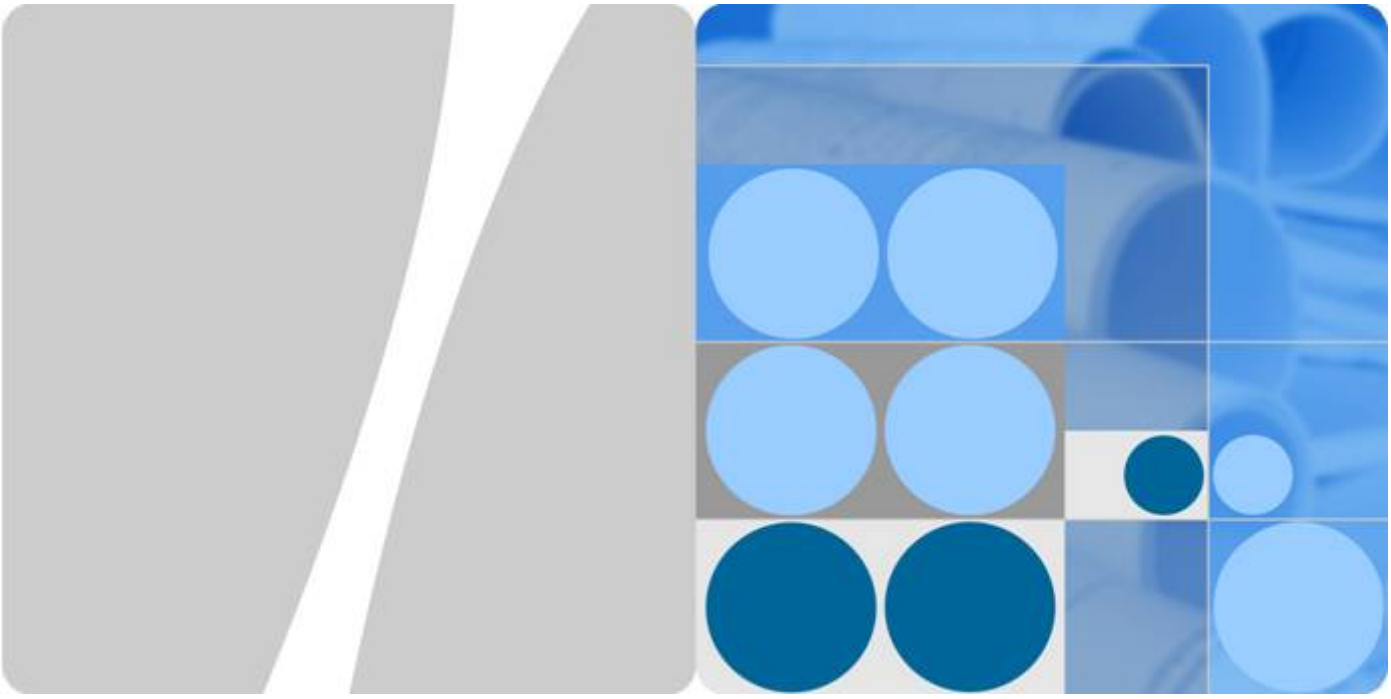


Distributed Cache Service

# User Guide

Issue 01

Date 2020-03-17

HUAWEI TECHNOLOGIES CO., LTD.

HUAWEI

### Trademarks and Permissions


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

### Notice

The purchased products, services and features are stipulated by the contract made between Huawei 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 Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

# Contents



# 1 Before You Start

## 1.1 Accessing and Using DCS

### Accessing DCS

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

- Using the management console

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

  For details on how to use the DCS console, see chapters from **3 Buying a DCS Instance** to **10 Managing Passwords**.

  DCS monitoring data is recorded by Cloud Eye. To view the monitoring metrics or configure alarm rules, go to the Cloud Eye console. For details, see **12.3 Viewing Metrics**.

  If you have enabled Cloud Trace Service (CTS), DCS instance operations are recorded by CTS. You can view the operations history on the CTS console. For details, see **13.2 Viewing Traces on the CTS Console**.

- Using APIs

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

**NOTICE**

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

## Using DCS

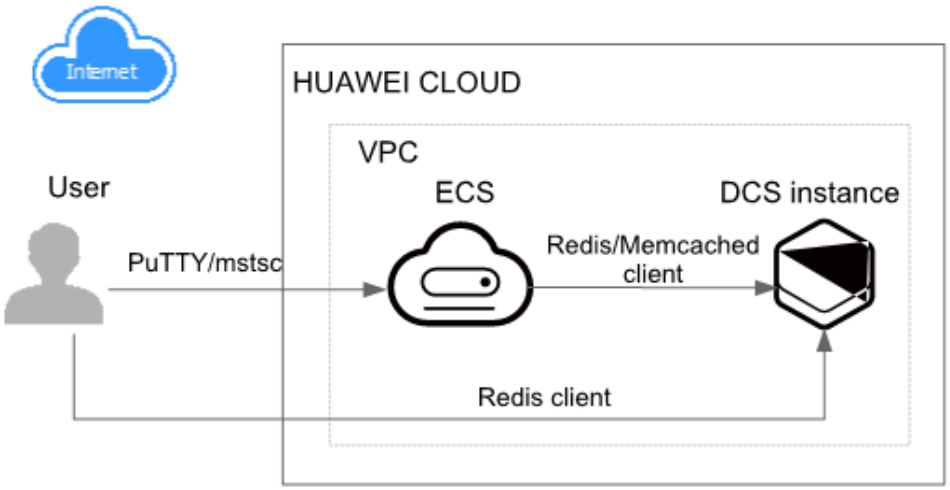
After purchasing a DCS instance, access it by referring to **4 Accessing a DCS Redis Instance**. Any client that is compatible with the open-source Redis or Memcached protocol can respectively access a DCS Redis or Memcached instance. After accessing a DCS instance, you can enjoy the fast read/write operations enabled by DCS.

**NOTICE**

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

For details on how to access a DCS instance, see **Figure 1-1**.

**Figure 1-1** Accessing a DCS instance



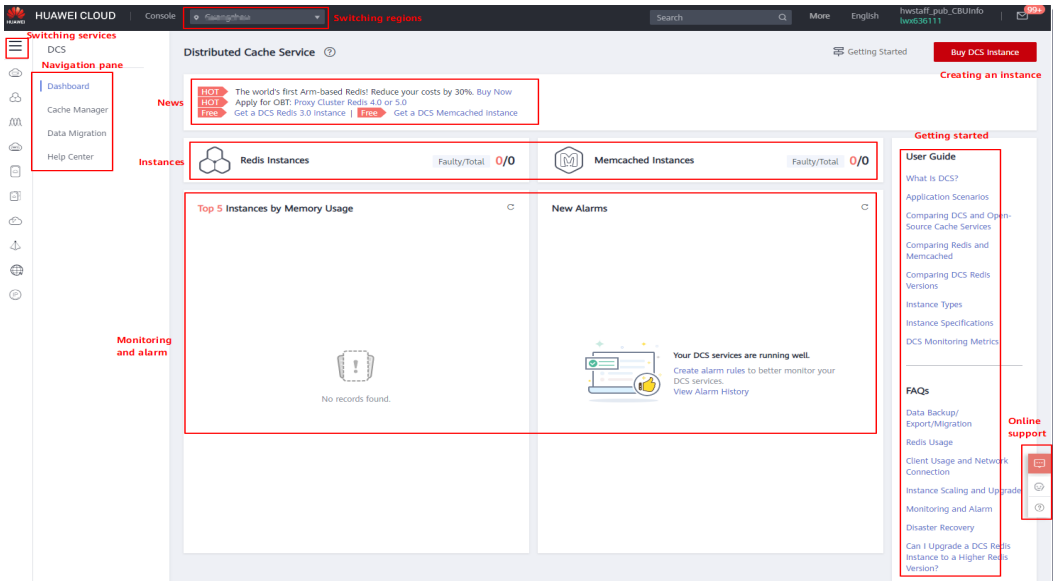

NOTE

- Currently, a DCS instance can be accessed over an internal network through an ECS that is in the same VPC as the DCS instance.
- If public access is enabled, a DCS Redis instance can be accessed through an elastic IP address (EIP) over a public network.

# 1.2 Using the DCS Console

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

**Figure 1-2** DCS console




- Switching regions

  You can switch to a region closer to your application.

- Switching services

  You can switch to consoles of other services, such as the VPC and Cloud Eye consoles.

- Creating an instance

  Click to buy DCS Redis or Memcached instances.

- Navigation pane

  This area provides access to operating DCS instances and migrating data.

- News

  This area informs you of the latest available features and special offers.

- Instances

  This area displays the total number of instances and the number of faulty instances of the current user.

- Monitoring and alarms

  This area displays instances with the highest memory usage. For details on how to view information about a specific instance, see **6.1 Viewing Instance Details**.

  You can create alarm rules for your instance. When an alarm is generated, you can handle it immediately. For details, see **12.4 Configuring Alarm Rules for Critical Metrics**.

- Getting started

  By clicking these links, you will be directed to the documentation to learn more about how to use DCS.

- Online support

  If you have any questions while using DCS, contact online support.

# 2 Permissions Management

## 2.1 Creating a User and Granting DCS Permissions

This chapter describes how to use **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.
- Grant only the permissions required for users to perform a specific task.
- 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.

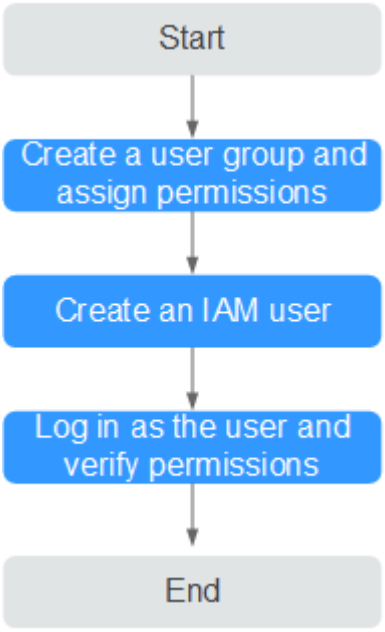
### Prerequisites

You are familiar with the permissions (see **Permissions Management**) supported by DCS and choose policies or roles according to your requirements. For the permissions of other services, see **Permissions Policies**.

### Process Flow

**Figure 2-1** Process of granting DCS permissions




1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console, and attach the **DCS ReadOnlyAccess** policy to the group.
2. **Create an IAM user**.

   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 by using the newly created user, and verify that the user only has read permissions for DCS.

## 2.2 DCS Custom Policies

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

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

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. The following section contains examples of common DCS custom policies.

NOTE

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

## Example Custom Policies

- Example 1: Allowing users to delete and stop a DCS instance

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

- Example 2: Denying DCS instance deletion

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

  The following method can be used if you need to assign permissions of the **DCS FullAccess** policy to a user but you want to prevent the user from deleting DCS instances. Create a custom policy for denying DCS instance deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on DCS instances except deleting DCS instances. The following is an example of a deny policy:

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

# 3 Buying a DCS Instance

## 3.1 Identifying Requirements

Before purchasing a DCS instance, identify your requirements and complete the following preparations:

1. Decide on the required cache engine.

   Choose a cache engine based on service requirements. The cache engine cannot be changed once the instance is created.

   - For more information about Redis and Memcached cache engines, see **DCS Overview**.
   - For more information about the differences between Redis and Memcached, see **Comparing Redis and Memcached**.

2. Decide on the required cache engine version.

   Perform this step if you choose Redis as the cache engine.

   Different Redis versions have different features. For details, see **Comparing DCS Redis Versions**.

3. Decide on the required instance type.

   DCS provides single-node, master/standby, Proxy Cluster, and Redis Cluster types of instances. Each type has its own architecture. For details about the instance architectures, see **DCS Instance Architecture**.

4. Decide on the required instance specification.

   Each specification specifies the maximum available memory, number of connections, and bandwidth. For details, see **DCS Instance Specifications**.

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

   Choose a region closest to your application to reduce latency. For example, if your application runs in Guangzhou, select the CN South-Guangzhou region.

   A region consists of multiple availability zones (AZs) with physically isolated power supplies and networks. Master/standby and cluster DCS instances can be deployed across AZs.

NOTE

- If a master/standby or cluster DCS instance is deployed across AZs, faults in an AZ do not affect cache nodes in other AZs. This is because when the master node is faulty, the standby cache node will automatically become the master node to provide services. Such deployment achieves better disaster recovery.
- Deploying a DCS instance across AZs slightly reduces network efficiency compared with deploying an instance within an AZ. Therefore, if a DCS instance is deployed across AZs, synchronization between master and standby cache nodes is slightly less efficient.

6. Decide whether public access is required.

   Currently, public access can be enabled only for DCS Redis 3.0 instances. You can configure whether to access the instance over public networks through SSL. For details, see **public access instructions**.

7. Check whether EIPs are available.

   Public access requires EIPs. If you decided that public access is not required, you do not need to purchase EIPs.

8. Decide whether backup policies are required.

   Currently, backup policies can be configured only for master/standby, Proxy Cluster, and Redis Cluster DCS instances. For details about backup and restoration, see **8.1 Overview**.

# 3.2 Preparing Required Resources

## Overview

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

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

## Required Resources

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

**Table 3-1** Dependency resources of a DCS instance

| Resource | Requirement | Operations |
| --- | --- | --- |
| VPC and subnet | Different DCS instances can use the same or different VPCs and subnets based on site requirements. Note the following when creating a VPC and subnet:<br>• The VPC and the DCS instance must be in the same region.<br>• 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**. |
| Security group<br>**NOTE**<br>Security groups are required only by DCS Redis 3.0 and Memcached instances. | Different DCS instances can use the same security group or different security groups. Note the following when creating a security group:<br>• Set **Template** to **Custom**.<br>• After a security group is created, retain the default inbound and outbound rules.<br>• To use DCS, you must add the security group rules described in **Table 3-2**. You can also add other rules based on site requirements. | For details on how to create a security group, see **Creating a Security Group**. For details on how to add rules to a security group, see **Adding a Security Group Rule**. |
| (Optional) EIP<br>**NOTE**<br>EIPs are supported only by DCS Redis 3.0 instances. | If you want to access DCS through a public network, apply for an EIP. | For details on how to apply for an EIP, see **Assigning an EIP**. |

**Table 3-2** Security group rules

| Direction | Protocol | Port | Source | Description |
| --- | --- | --- | --- | --- |
| Inbound | TCP | 36379 | 0.0.0.0/0 | Access a DCS Redis instance (with SSL encryption enabled) through a public network. |

| Direction | Protocol | Port | Source | Description |
| --- | --- | --- | --- | --- |
| Inbound | TCP | 6379 | 0.0.0.0/0 | Access a DCS Redis instance (with SSL encryption disabled) through a public network. |
| Inbound | TCP | 6379 | 0.0.0.0/0 | Access a DCS Redis instance in a private network. (SSL encryption is not supported.) |
| Inbound | TCP | 11211 | 0.0.0.0/0 | Access a DCS Memcached instance through a public network. (SSL encryption is not supported.) |

# 3.3 Buying a DCS Redis Instance

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

NOTE

DCS for Redis 3.0 is about to become unavailable and is no longer sold in some regions. You can use DCS for Redis 4.0 or 5.0 instead.

## Prerequisites

To achieve fine-grained management of your HUAWEI CLOUD resources, create IAM user groups and users and grant specified permissions to the users. For details, see **2 Permissions Management**.

## Buying a DCS Redis Instance

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

**Step 3** Click **Buy DCS Instance**.

**Step 4** Specify **Billing Mode**.

**Step 5** Select a region closest to your application to reduce latency and accelerate access.

**Step 6** Specify the following instance parameters based on the information collected in **3.1 Identifying Requirements**.

1. **Cache Engine**:

Select **Redis**.

2. **Version**:

   Currently supported versions: Redis 3.0, 4.0, and 5.0, and Redis 6.0 professional edition.

3. Set **Instance Type** to **Single-node**, **Master/Standby**, **Proxy Cluster**, **Redis Cluster**, or **Read/Write splitting**.
4. Set **CPU Architecture** to **x86** or **Arm**.
5. Set **Replicas**. The default value is **2**.

   This parameter is displayed only when you select Redis 4.0 or Redis 5.0 and the instance type is master/standby or Redis Cluster.

6. If **Redis Cluster** is selected, the **Sharding** parameter is displayed.

   **Use default**: Use the default sharding specifications.

   **Customize**: Customize the size of each shard and then select corresponding instance specifications.

7. Select an AZ.

   NOTE

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

   If the instance type is master/standby, Proxy Cluster, or Redis Cluster, **Standby AZ** is displayed. Select a standby AZ for the standby node of the instance.

8. **Instance Specification**:

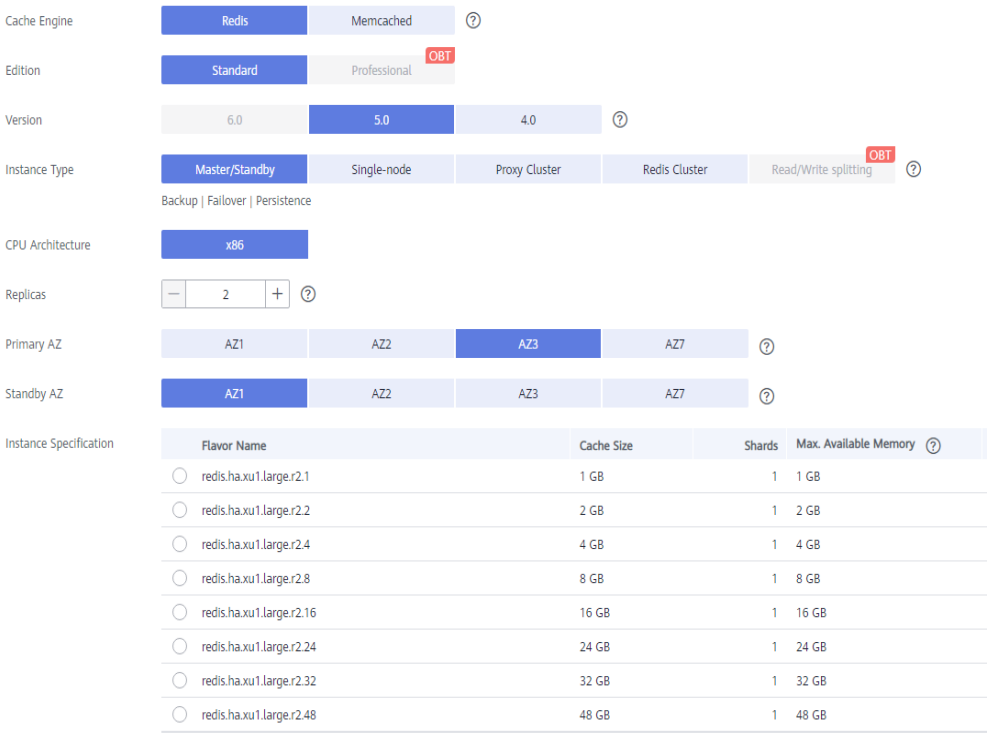
   The remaining quota is displayed on the console.

   To increase quota, click **Increase quota** below the specifications. On the displayed page, fill in a quota application form and click **Submit**.

   **Figure 3-1** shows the instance parameter settings.

**Figure 3-1** Buying a DCS Redis instance




**Step 7** Configure the instance network parameters.

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

   All Redis versions support automatically-assigned IP address and manually-specified IP addresses. You can manually specify a private IP address for your instance as required.

   For Redis 4.0 or 5.0, 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.

   For Redis 6.0 enterprise edition, 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.

   For Redis 3.0, you cannot customize a port. The default port 6379 will be used.
3. Select a security group.

   A security group is a set of rules that control access to ECSs. It provides access policies for mutually trusted ECSs with the same security protection requirements in the same VPC.

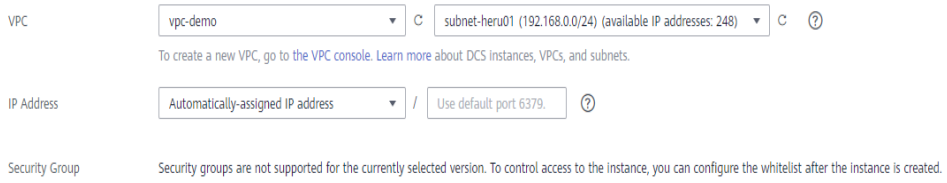
   DCS for Redis 4.0 and 5.0 are based on VPC endpoints and do not require security groups.

   Redis 6.0 professional edition is also based on VPC endpoints and does not require security groups.

   If port 6379 is not enabled for the selected security group, the **Enable port 6379** check box is displayed and selected by default, indicating that after the instance is created, port 6379 will be enabled for the selected security group. If port 6379 is not enabled for the selected security group, connections to the instance may fail.

**Figure 3-2** Configuring instance network parameters




**Step 8** Set the instance password.

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

  NOTE

  - Password-free access carries security risks. Exercise caution when selecting this mode.
  - If you are to enable public access, you must select the password-protected mode and set a password.
  - After creating a DCS Redis instance to be accessed in password-free mode, you can set a password for it by using the password reset function. For details, see **10.4 Changing Password Settings for DCS Redis Instances**.

- **Password** and **Confirm Password**: These parameters indicate the password of accessing the DCS Redis instance, and are displayed only when **Password Protected** is set to **Yes**.

  NOTE

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

**Step 9** Specify the required duration and quantity.

NOTE

After enabling public access, you can only create one DCS Redis instance at time.

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

1. Specify **Name** and **Description**.

   When you create only one instance at a time, the value of **Name** must be a string consisting of 4 to 64 characters. When you create more than one instance at a time, the value of **Name** must be a string consisting of 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**.

2. Configure public access.

   You can enable or disable public access. Currently, only DCS Redis 3.0 instances support public access.

   - A DCS Redis instance with public access enabled is accessed using an EIP. After you enable public access, **Elastic IP Address** and **SSL** are displayed. Select an EIP or click **View Elastic IP** to view or create EIPs.
   - **SSL**: An indicator of whether to enable SSL encryption during public access to your DCS Redis instance.

When public access is enabled, **SSL** is enabled by default. This parameter cannot be modified once the instance is created with public access enabled.

If you want to change the SSL setting after the instance is created, disable public access first to change the SSL setting.

3. Specify **Enterprise Project**.
4. Specify backup and restoration policies.

   This parameter is displayed only when the instance type is master/standby or cluster. For more information on how to configure a backup policy, see **Backing Up and Restoring Instances**.
5. Rename critical commands.

   If Redis 4.0 or 5.0, or Redis 6.0 professional edition is selected, the **Command Renaming** parameter is displayed. Currently, you can only rename the **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, and **HGETALL** 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. 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 create new tags by specifying **Tag key** and **Tag value**.

   Up to 10 tags can be added to each DCS instance. For details about the requirements on tags, see **7.7 Managing Tags**.

**Step 11** Click **Next**.

The displayed page shows the instance information you have specified.

**Step 12** Confirm the instance information and click **Submit**.

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

**----End**

## Video Guide

Watch the following video about creating a DCS Redis instance on the DCS console.

**Creating a DCS Instance**

# 3.4 Buying a DCS Memcached Instance (Unavailable Soon)

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

NOTE

DCS for Memcached is about to become unavailable and is no longer sold in some regions. You can use DCS for Redis 4.0 or 5.0 instead.

## Prerequisites

To achieve fine-grained management of your HUAWEI CLOUD resources, create IAM user groups and users and grant specified permissions to the users. For details, see **2 Permissions Management**.

## Buying a DCS Memcached Instance

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click **Buy DCS Instance**.

**Step 5** Specify **Billing Mode**.

**Step 6** Select a region closest to your application to reduce latency and accelerate access.

**Step 7** Specify the following instance parameters based on the information collected in **3.1 Identifying Requirements**.

1. Set **Cache Engine** to **Memcached**.
2. Set **Instance Type** to either **Single-node** or **Master/Standby**.
3. Select an AZ.

   NOTE

   To accelerate access, deploy your instance and your application in the same AZ. To ensure data reliability, deploy them in different AZs.

   If the instance type is master/standby, **Standby AZ** is displayed. Select a standby AZ for the standby node of the instance.
4. Specify **Instance Specification**.

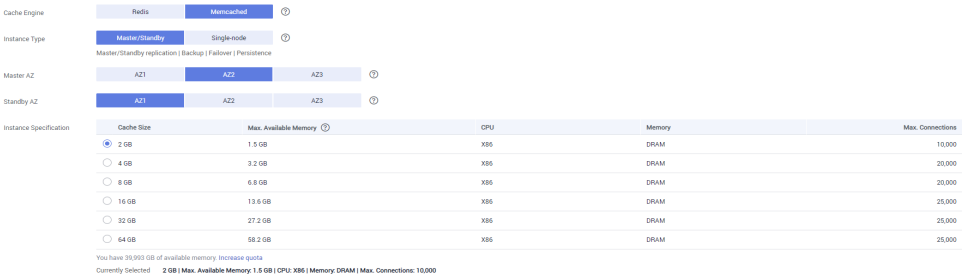
   The quota displayed on the console prevails.

   To increase quota, click **Increase quota** below the specifications. On the displayed page, fill in a quota application form and click **Submit**.

Figure 3-3 shows the instance parameter settings.

**Figure 3-3** Buying a DCS Memcached Instance




**Step 8** Configure the instance network parameters.

1. For **VPC**, select a VPC, subnet, and specify the IP address.

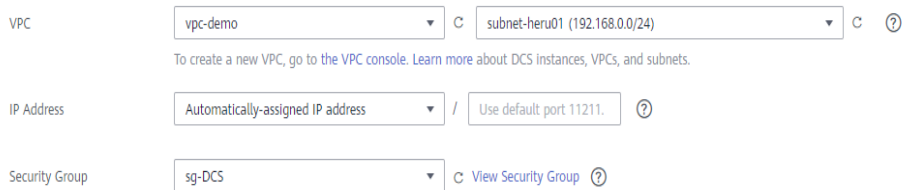
   You can choose to obtain an automatically assigned IP address or manually specify an IP address that is available in the selected subnet.

2. Select a security group.

   A security group is a set of rules that control access to ECSs. It provides access policies for mutually trusted ECSs with the same security protection requirements in the same VPC.

   If port 11211 is not enabled for the selected security group, the **Enable port 11211** check box is displayed and selected by default, indicating that after the instance is created, port 11211 will be enabled for the selected security group. If port 11211 is not enabled for the selected security group, connections to the instance may fail.

**Figure 3-4** Configuring instance network parameters




**Step 9** 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 DCS Memcached instance in password-protected mode, you can reset the password or change it into password-free mode. For details, see **10.5 Changing Password Settings for DCS Memcached Instances**.
  - If password-free access is disabled, DCS Memcached instances must be accessed using the Memcached binary protocol and through SASL authentication.

- Username required for accessing the new DCS instance. The username must meet the following requirements.

  NOTE

  This parameter is displayed only when **Password Protected** is set to **Yes**.

- **Password** and **Confirm Password**: These parameters indicate the password of accessing the DCS Memcached 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 Memcached instance. Keep your instance password secure and change it periodically.

**Step 10** Specify the required duration and quantity.

**Step 11** Click **More Settings** to display more configurations, including backup policy and instance tags.

1. Specify **Name** and **Description**.

   When you create only one instance at a time, the value of **Name** must be a string consisting of 4 to 64 characters. When you create more than one instance at a time, the value of **Name** must be a string consisting of 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**.

2. Specify **Enterprise Project**.
3. Specify backup and restoration policies.

   This parameter is displayed only when the instance type is master/standby. For more information on how to configure a backup policy, see **Backing Up and Restoring Instances**.

4. Specify the maintenance time window.

   Choose a window for DCS O&M personnel to perform maintenance on your instance. Time windows 22:00–02:00, 02:00–06:00, 06:00–10:00, 10:00–14:00, 14:00–18:00, and 18:00–22:00 are available for selection.

5. 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 or create predefined tags.
   - You can also create new tags by specifying **Tag key** and **Tag value**.

   Up to 10 tags can be added to each DCS instance. For details about the requirements on tags, see **7.7 Managing Tags**.

**Step 12** Click **Next**.

The displayed page shows the instance information you have specified.

**Step 13** Confirm the instance information and click **Submit**.

**Step 14** After the new DCS instance has been created, return to the **Cache Manager** page to view and manage your DCS instances.

1. It takes 5 to 15 minutes to create a DCS instance.

2. After a DCS instance has been successfully created, it enters the **Running** state by default.

**----End**

# 4 Accessing a DCS Redis Instance

## 4.1 Restrictions

You can access a DCS instance through any Redis client.

- To access a DCS Redis instance through a client on an ECS in the same VPC as the instance, note that:
  - An ECS and a DCS instance can communicate with each other only when they belong to the same VPC.
    - Redis 3.0: The instance and the ECS must either be configured with the same security group or use different security groups but can communicate with each other as configured by the security group rules.
    - Redis 4.0 and 5.0, and Redis 6.0 professional edition: The IP address of the ECS must be on the whitelist of the DCS instance.
  - If the ECS and DCS Redis instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**
- Before accessing a DCS Redis 3.0 instance over public networks, ensure that:

  Security group rules have been correctly configured for the instance. If SSL encryption is disabled, allow public access through port 6379. If SSL encryption is enabled, allow public access through port 36379. For details, see **How Do I Configure a Security Group?**
- If the client and the DCS Redis instance are not in the same region, the instance domain name cannot be resolved across regions and the instance cannot be accessed at its domain name address. 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 Public Access to a DCS Redis 3.0 Instance (Unavailable)

# 4.2.1 Step 1: Check Whether Public Access Is Supported

You can access a DCS Redis 3.0 instance over public networks. In comparison with intra-VPC access, public access may bring packet loss, jitter, and higher latency. Therefore, you are advised to enable public access only during the service development and testing phases.

Before connecting to a DCS instance over public networks, check whether the instance supports public access.

- Redis 3.0

  Currently, **only DCS Redis 3.0 instances support public access**. You can enable or disable public access.

- Redis 4.0 and 5.0

  **Public access is not supported by DCS Redis 4.0 and 5.0 instances**. If public access is required for a single-node, master/standby, or Proxy Cluster instance, use Nginx to redirect connections through an ECS configured with the same VPC and security group as the DCS instance. For details, see **Using Nginx to Access DCS Redis 4.0 or 5.0 Instances over Public Networks**.

  Redis Cluster instances cannot be accessed over public networks.

- Memcached

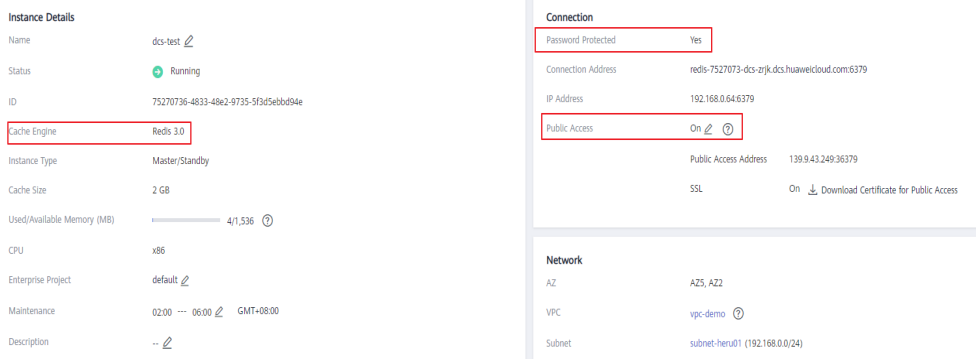
  **Public access is not supported by DCS Memcached instances**. The ECS that serves as a client and the DCS instance that the client will access must belong to the same VPC. In the application development and debugging phase, you can also use an SSH agent to access DCS Memcached instances in the local environment.

## Procedure

On the **Basic Information** page of the instance, check the following parameter settings:

- **Cache Engine**: Must be **Redis 3.0**. If not, public network access is not supported.
- **Password Protected**: Must be **Yes**. If not, enable password protection for the instance.
- **Public Access**: Must be **On**. If not, enable public access by referring to **4.2.2 Step 2: Enable Public Access for a DCS Redis Instance**.

**Figure 4-1** Checking the cache engine version, password protection, and public access



## FAQs

- What can I do if the public access button is grayed out when the instance is not password-protected?

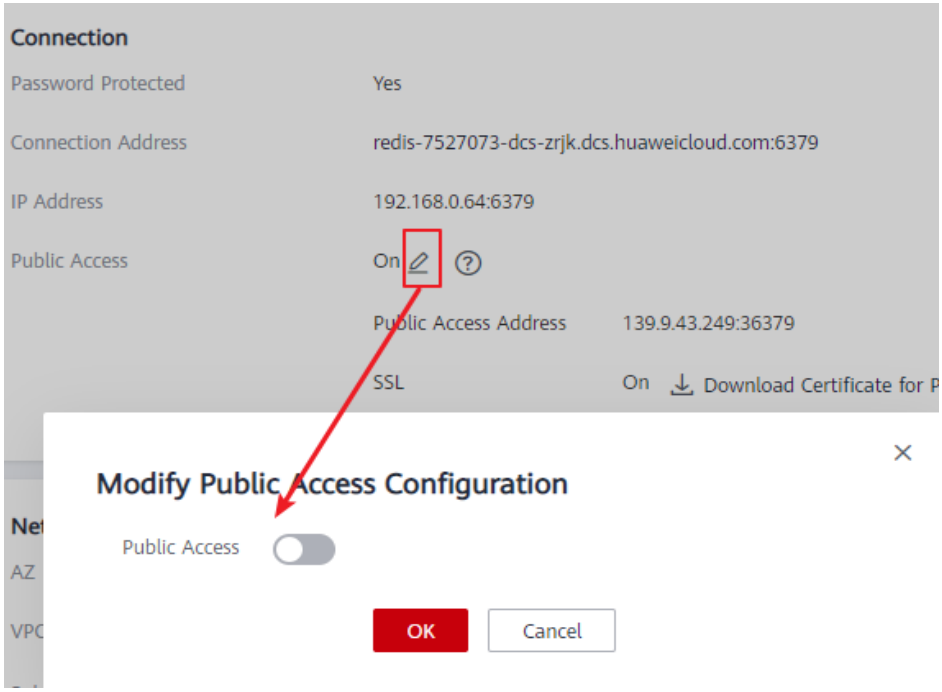
  In the upper right corner of the **Basic Information** page, choose **More** > **Reset Password**. After the password is reset, the **Password Protected** parameter changes to **Yes**. The public access button can be clicked now.

- How do I disable SSL encryption when public access has been enabled?

  Solution: SSL encryption is enabled by default when you enable public access. **Once public access has been enabled, the SSL setting cannot be changed**. If you need to disable SSL, do as follows:

  a. Disable public access as shown in the following figure.

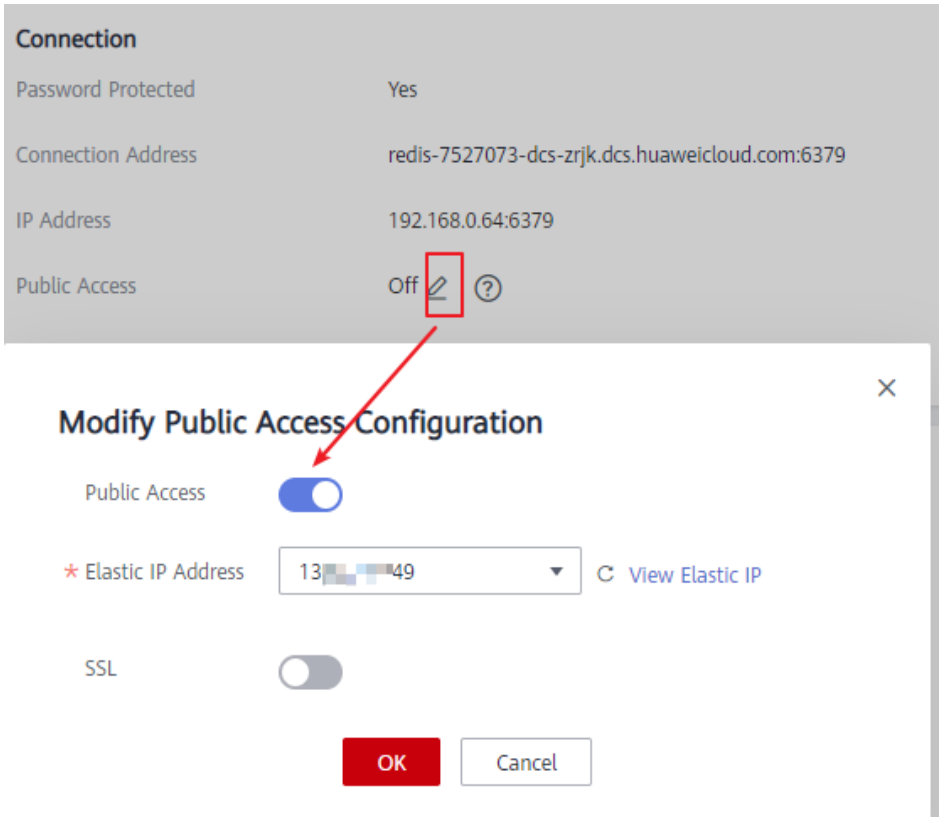
  **Figure 4-2** Disabling public access

  Wait until public access has been disabled.

  b. Re-enable public access. During this step, disable SSL as shown in the following figure.

**Figure 4-3** Enabling public access while disabling SSL




## 4.2.2 Step 2: Enable Public Access for a DCS Redis Instance

If public access has been enabled for the instance, skip this section.

If public access is not enabled, follow the instructions in this section. You can enable or disable SSL encryption when enabling public access.

NOTE

- Before accessing a DCS instance through a public network (with SSL encryption), download a CA certificate to verify the certificate of the instance for security purposes.
- When accessing a DCS instance through a public network (without SSL encryption), access the EIP and port 6379 of the instance. You do not need to download certificates or install Stunnel on your client.
- You are advised to enable SSL to encrypt the data transmitted between your Redis client and DCS instance to prevent data leakage.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of the DCS Redis instance you want to configure. A page with basic information about the DCS instance is displayed.

**Step 5** Click  on the right of **Public Access**.

**Step 6** Click  .

**Step 7** Select an EIP from the **Elastic IP Address** drop-down list.

If no EIPs are available, click **View Elastic IP** to create an EIP on the network console. After an EIP is created, click the refresh button on the right of **Elastic IP Address** to select the new EIP.

**Step 8** (Optional) Enable or disable SSL as required.

You are advised to enable SSL to encrypt the data transmitted between your Redis client and DCS instance to prevent data leakage.

**This parameter cannot be modified once the instance is created with public access enabled.** If you want to change the SSL setting after the instance is created, disable public access first to change the SSL setting.

**Step 9** Click **OK**.

It takes 1 to 2 minutes to enable public access. Please wait.

You will be automatically redirected to the **Background Tasks** page, where the progress of the current task is displayed. If the task status is **Succeeded**, public access is successfully enabled.

**----End**

## 4.2.3 Step 3: Access a DCS Redis Instance in Windows

This section describes how to access a DCS Redis 3.0 instance over a public network by using redis-cli in Windows.

Public access helps R&D personnel establish local environment for development or testing, improving development efficiency. However, in the production environment (official environment), access a DCS Redis instance through a VPC to ensure efficient access.

### Prerequisites

Before using redis-cli to access a DCS Redis instance over a public network, ensure that:

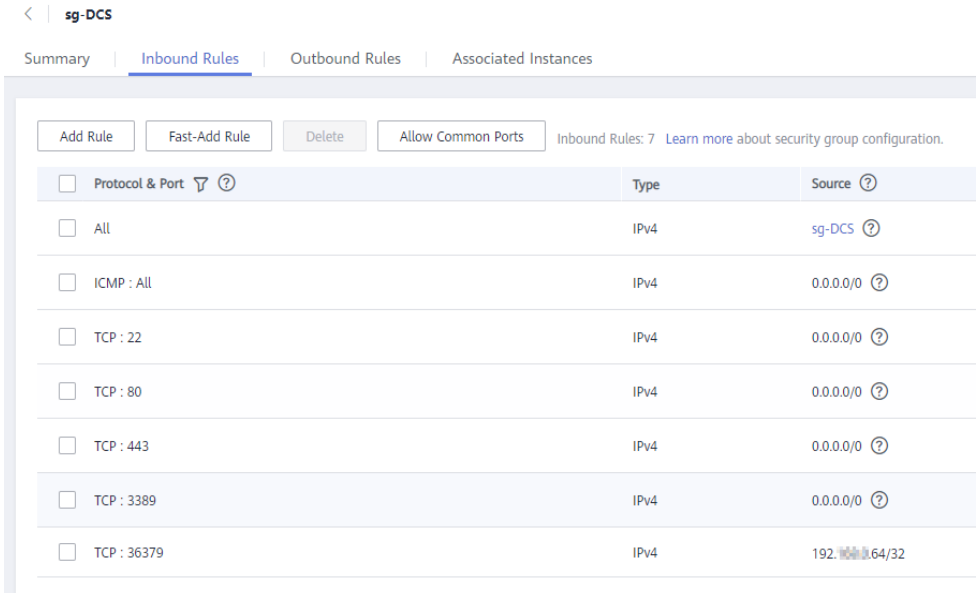
- The instance version is Redis 3.0 and public access has been enabled.
- If certificates are required for accessing the DCS instance, download the certificate from the DCS instance details page. For details, see **6.1 Viewing Instance Details**.

## Connecting to Redis with SSL Encryption

**Step 1** Ensure that the security group rule allows public access through port 36379.

When SSL encryption is enabled, allow public access through port 36379 and install the Stunnel client.

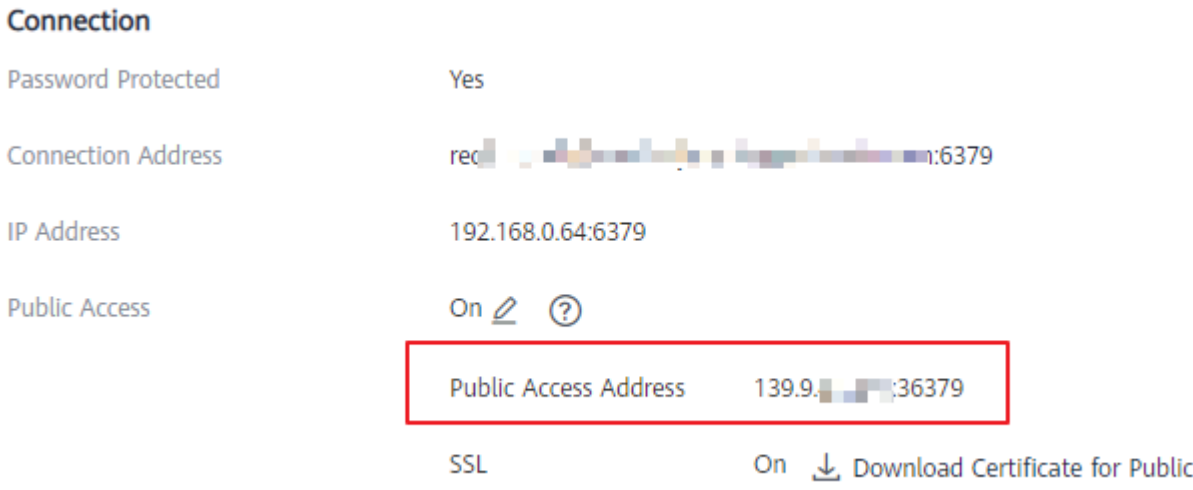
**Figure 4-4** Security group rule (port 36379)




**Step 2** Obtain the public access address and the certificates of the instance on the instance **Basic Information** page.

- The public access address is displayed in the **Connection** section.
- The certificates can be downloaded by clicking **Download Certificate for Public Access** in the **Connection** section. After decompression, you will obtain **dcs-ca.cer** (the public key certificate in binary format) and **dcs-ca-bundle.pem** (the certificate file in text format).

**Figure 4-5** Viewing the public access address (SSL enabled; port 36379)



**Step 3** Download the latest Windows Stunnel installation package (for example, **stunnel-5.44-win32-installer.exe**) from **https://www.stunnel.org/downloads.html** to the local Windows device.

**Step 4** Run the Stunnel installation program and install the Stunnel client.

**Step 5** Configure the Stunnel client: Right click  on the taskbar and choose **Edit Configuration**. Add the following configuration and then save and exit.

```
[redis-client]
client = yes
CAfile = D:\tmp\dcs\dcs-ca.cer
accept = 8000
connect = {public access address}
```

In the configuration:

- **client**: indicates Stunnel. The fixed value is **yes**.
- **CAfile**: specifies a CA certificate, which is optional. If a CA certificate is required, download and decompress the certificate **dcs-ca.cer** as instructed in **Step 2**. If it is not required, delete this parameter.
- **accept**: specifies the user-defined listening port number of Stunnel. Specify this parameter when accessing a DCS instance by using a Redis client.
- **connect**: specifies the service address and port number of Stunnel. Set this parameter to the instance public access address obtained in **Step 2**.

When SSL encryption is enabled, the configuration is similar to the following:

```
[redis-client]
client = yes
CAfile = D:\tmp\dcs\dcs-ca.cer
accept = 8000
connect = 49.**.**.211:36379
```

**Step 6** Right-click  on the taskbar and choose **Reload Configuration**.

**Step 7** Open the CLI tool **cmd.exe** and run the following command to check whether 127.0.0.1:8000 is being listened:

**netstat -an |find "8000"**

Assume that port **8000** is configured as the listening port on the client.

If **127.0.0.1:8000** is displayed in the returned result and its status is **LISTENING**, the Stunnel client is running properly. When the Redis client connects to the address **127.0.0.1:8000**, Stunnel will forward requests to the DCS Redis instance.

**Step 8** Access the DCS Redis instance.

1. Obtain and decompress the Redis client installation package.

   **Download** the Windows Redis client installation package.

2. Open the CLI tool **cmd.exe** and run commands to go to the directory where the decompressed Redis client installation package is saved.

   For example, to go to the **D:\redis-64.3.0.503** directory, run the following commands:

   **D:**

   **cd D:\redis-64.3.0.503**

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

   **redis-cli -h 127.0.0.1 -p 8000 -a <password>**

   **CAUTION**

   In the preceding command:

   - The address following **-h** indicates the address of the Stunnel client, which is **127.0.0.1**.
   - The port following **-p** is the listening port of the Stunnel client, which has been configured in the **accept** field in **Step 5**. **8000** is used an example here.

   Do not use the public access address and port displayed on the console for the **-h** and **-p** parameters.

   *<password>* 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:

   ```
   127.0.0.1:8000>
   ```

   After you enter **info**, the DCS instance information is returned. If no information is returned or the connection is interrupted, right-click the Stunnel icon on the taskbar and choose **Show Log Window** from the shortcut menu to show logs of Stunnel for cause analysis.

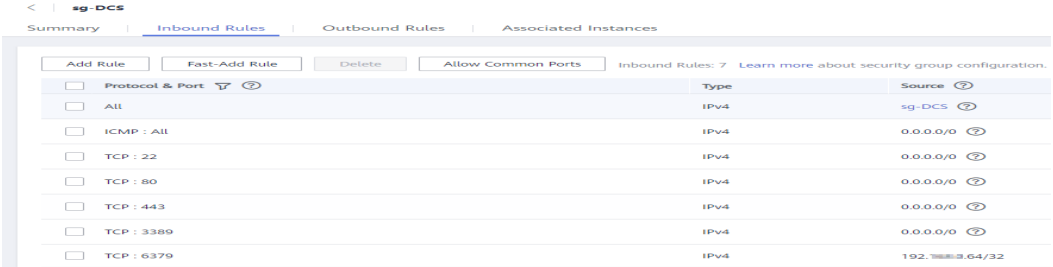
**----End**

## Connecting to Redis Without SSL Encryption

**Step 1** Ensure that the security group rule allows public access through port 6379.

When SSL encryption is disabled, the instance public access address can be accessed only if access through port 6379 is allowed.

**Figure 4-6** Security group rule (port 6379)

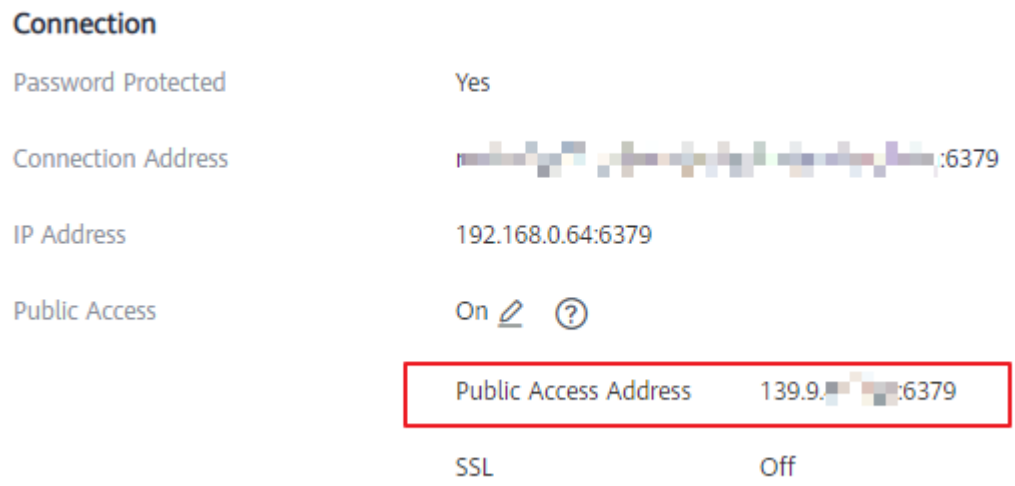



**Step 2** Obtain the public access address of the instance.

The public access address is displayed in the **Connection** section.

**Figure 4-7** Viewing the public access address (SSL disabled; port 6379)




**Step 3** Download the Redis client installation package to the local Windows device and decompress the package.

**Download** the Windows Redis client installation package.

**Step 4** Open the CLI tool **cmd.exe** and run commands to go to the directory where the decompressed Redis client installation package is saved.

For example, to go to the **D:\redis-64.3.0.503** directory, run the following commands:

**D:**

**cd D:\redis-64.3.0.503**

**Step 5** Run the following command to access the chosen DCS Redis instance:

**redis-cli -h** *{public network access IP}* **-p 6379 -a** *<password>*

In this command, *{public network access IP}* indicates the IP address of the DCS Redis instance obtained in **Step 2**. *<password>* 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:

```
139.**.**.175:6379>
```

Enter **info** and the DCS instance information will be returned.

**----End**

## Troubleshooting

- **Symptom**: "Error: Connection reset by peer" is displayed or a message is displayed indicating that the remote host forcibly closes an existing connection.

  **Possible cause 1**: The security group is incorrectly configured. You need to enable port **36379** or **6379**.

**Possible cause 2**: SSL encryption has been enabled, but Stunnel is not configured during connection. The IP address displayed on the console was used for connection. In this case, strictly follow the instructions provided in **Connecting to Redis with SSL Encryption**.

- For more information about Redis connection failures, see **Troubleshooting Redis Connection Exceptions**.

## 4.2.4 Step 3: Access a DCS Redis Instance in Linux

This section describes how to access a DCS Redis 3.0 instance over a public network by using redis-cli in Linux.

Public access helps R&D personnel establish local environment for development or testing, improving development efficiency. However, in the production environment (official environment), access a DCS Redis instance through a VPC to ensure efficient access.

### Prerequisites

Before using redis-cli to access a DCS Redis instance over a public network, ensure that:

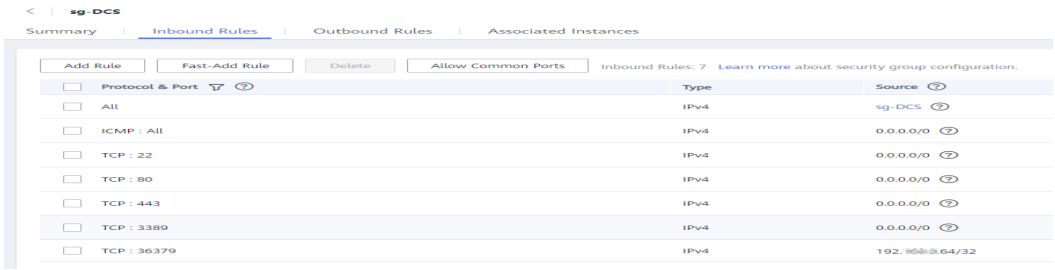
- The instance version is Redis 3.0 and public access has been enabled.
- If certificates are required for accessing the DCS instance, download the certificate from the DCS instance details page. For details, see **6.1 Viewing Instance Details**.

### Connecting to Redis with SSL Encryption

**Step 1** Ensure that the security group rule allows public access through port 36379.

When SSL encryption is enabled, port 36379 must be allowed to be accessed by external addresses. Ensure that the Stunnel client has been installed.

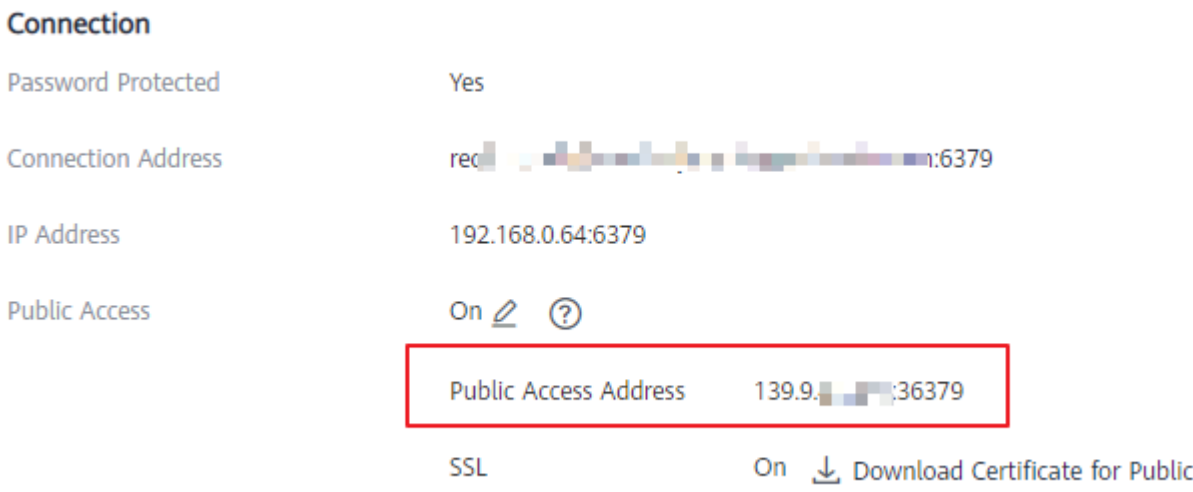
**Figure 4-8** Security group rule (port 36379)




**Step 2** Obtain the public access address and the certificates of the instance on the instance **Basic Information** page.

- The public access address is displayed in the **Connection** section.
- The certificates can be downloaded by clicking **Download Certificate for Public Access** in the **Connection** section. After decompression, you will obtain **dcs-ca.cer** (the public key certificate in binary format) and **dcs-ca-bundle.pem** (the certificate file in text format).

**Figure 4-9** Viewing the public access address (SSL enabled; port 36379)




**Step 3** Log in to the local Linux device.

**Step 4** Install the Stunnel client.

Use either of the following methods to install Stunnel.

NOTE

Installation methods **apt** and **yum** are recommended. Any common Linux OSs should support at least one of these installation methods.

- **apt-get** method:

  **apt-get** is used to manage DEB software packages and applicable to Debian OSs such as Ubuntu. Run the following command to install Stunnel:

  **apt install stunnel** or **apt-get install stunnel**

  If you cannot find Stunnel after running the command, run the **apt update** command to update the configuration and then install Stunnel again.

- **yum** method:

  **yum** is used to manage RPM software packages and applicable to OSs such as Fedora, CentOS, and Red Hat. Run the following command to install Stunnel:

  **yum install stunnel**

**Step 5** Open the Stunnel configuration file **stunnel.conf**.

- If Stunnel is installed using **apt-get**, the configuration file is stored at the **/etc/stunnel/stunnel.conf** directory by default.

  If the **/etc/stunnel/stunnel.conf** directory does not exist or no configuration file exists in the directory, add a directory or configuration file.

- If Stunnel is installed using **yum**, the configuration file is stored at the **/usr/local/stunnel/stunnel.conf** directory by default.

  If the **/etc/stunnel/stunnel.conf** directory does not exist or no configuration file exists in the directory, add a directory or configuration file.

NOTE

- If you are not sure where to store the configuration file, enter the **stunnel** command after the installation to view the directory for storing the configuration file.
- The configuration file can be stored in any directory. Specify this configuration file when starting Stunnel.

**Step 6** Add the following content to the configuration file **stunnel.conf**, and then save and exit.

```
debug = 4
output = /var/log/stunnel.log
sslVersion = all
[redis-client]
client = yes
accept = 8000
connect = {public access address}
CAfile = /etc/stunnel/dcs-ca.cer
```

In the configuration:

- **client**: indicates Stunnel. The fixed value is **yes**.
- **CAfile**: specifies a CA certificate, which is optional. If a CA certificate is required, download and decompress the certificate **dcs-ca.cer** as instructed in **Step 2**. If it is not required, delete this parameter.
- **accept**: specifies the user-defined listening port number of Stunnel. Specify this parameter when accessing a DCS instance by using a Redis client.
- **connect**: specifies the forwarding address and port number of Stunnel. Set this parameter to the instance public access address obtained in **Step 2**.

The following is a configuration example:

```
[redis-client]
client = yes
CAfile = D:\tmp\dcs\dcs-ca.cer
accept = 8000
connect = 49.**.**.211:36379
```

**Step 7** Run the following commands to start Stunnel:

**stunnel /*{customdir}*/stunnel.conf**

In the preceding command, **{*customdir*}** indicates the customized storage directory for the **stunnel.conf** file described in **Step 5**. The following is a command example:

**stunnel /etc/stunnel/stunnel.conf**

NOTE

For the Ubuntu OS, run the **/etc/init.d/stunnel4 start** command to start Stunnel. The service or process name is **stunnel4** for the Stunnel 4.x version.

After starting the Stunnel client, run the **ps -ef|grep stunnel** command to check whether the process is running properly.

**Step 8** Run the following command to check whether Stunnel is being listened:

**netstat -plunt |grep *8000*|grep "LISTEN"**

**8000** indicates the user-defined listening port number of Stunnel configured in the **accept** field in **Step 6**.

If a line containing the port number **8000** is displayed in the returned result, Stunnel is running properly. When the Redis client connects to the address **127.0.0.1:8000**, Stunnel will forward requests to the DCS Redis instance.

**Step 9** Access the DCS Redis instance.

1. Log in to the local Linux device.
2. Run the following command to download the source code package of your Redis client from **http://download.redis.io/releases/redis-5.0.8.tar.gz**:

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

   NOTE

   You can also install the Redis client by running the following yum or apt command:
   - **yum install redis**
   - **apt install redis-server**
3. Run the following command to decompress the source code package of your Redis client:

   **tar -xzf redis-5.0.8.tar.gz**
4. Run the following commands to go to the **redis-3.0.7** directory and compile the source code of your Redis client:

   **cd redis-5.0.8**

   **make**
5. Run the following commands to access the chosen DCS Redis instance:

   **cd src**

   **./redis-cli -h 127.0.0.1 -p *8000***

   CAUTION

   In the preceding command:
   - The address following **-h** indicates the address of the Stunnel client, which is **127.0.0.1**.
   - The port following **-p** is the listening port of the Stunnel client, which has been configured in the **accept** field in **Step 6**. **8000** is used an example.

   Do not use the public access address and port displayed on the console for the **-h** and **-p** parameters.
6. Enter the password. You can read and write cached data only after the password is verified.

   **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.

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

   ```
   OK
   127.0.0.1:8000>
   ```

**----End**

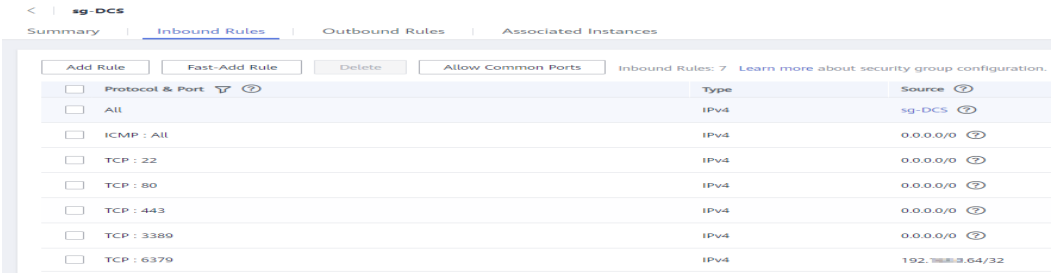
  

## Connecting to Redis Without SSL Encryption

**Step 1** Ensure that the security group rule allows public access through port 6379.

When SSL encryption is disabled, the instance public access address can be accessed only if access through port 6379 is allowed.

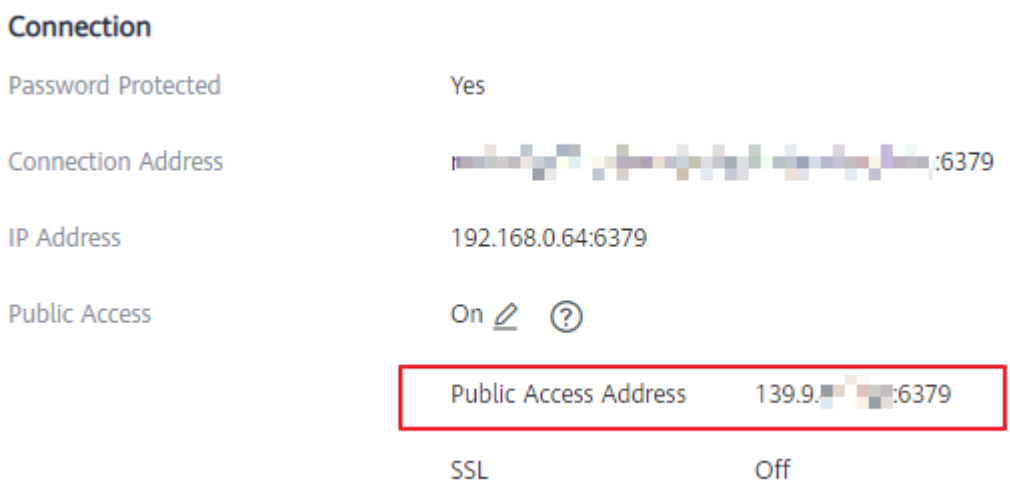
**Figure 4-10** Security group rule (port 6379)




**Step 2** Obtain the public access address of the instance.

The public access address is displayed in the **Connection** section of the instance **Basic Information** page.

**Figure 4-11** Viewing the public access address (SSL disabled; port 6379)




**Step 3** Log in to the local Linux device.

**Step 4** Run the following command to download the source code package of your Redis client from **http://download.redis.io/releases/redis-5.0.8.tar.gz**:

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

NOTE

You can also install the Redis client by running the following yum or apt command:

- **yum install redis**
- **apt install redis-server**

**Step 5** Run the following command to decompress the source code package of your Redis client:

**tar -xzf redis-5.0.8.tar.gz**

**Step 6** Run the following commands to go to the **redis-3.0.7** directory and compile the source code of your Redis client:

**cd redis-5.0.8**

**make**

**Step 7** Run the following commands to access the chosen DCS Redis instance:

**cd src**

**./redis-cli -h *{public access address}* -p 6379**

Replace {*public access address*} with the address obtained in **Step 2**. For example:

**./redis-cli -h 49.**.**.211 -p 6379**

**Step 8** Enter the password. You can read and write cached data only after the password is verified.

**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.

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

```
OK
49.**.**.211:6379>
```

**----End**

## Troubleshooting

- Symptom: "Error: Connection reset by peer" is displayed.

  **Possible cause**: The security group is incorrectly configured. You need to enable port **36379** or **6379**.

- When redis-cli is used to connect to an instance, the following message is displayed indicating that the remote host forcibly closes an existing connection.

  **Possible cause**: SSL encryption has been enabled, but Stunnel is not configured during connection. The IP address displayed on the console was used for connection. In this case, strictly follow the instructions provided in **Connecting to Redis with SSL Encryption**.

- For more information about Redis connection failures, see **Troubleshooting Redis Connection Exceptions**.

# 4.3 Multi-Language Access to a DCS Redis Instance

## 4.3.1 Access Using redis-cli

Access a DCS Redis instance through redis-cli on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

For more information on how to access a DCS Redis instance over public networks, see **4.2.3 Step 3: Access a DCS Redis Instance in Windows**.

NOTE

- Redis 3.0 does not support port customization and allows only port 6379. For Redis 4.0 and 5.0, you can specify a port or use the default port 6379. The following uses the default port 6379. If you have specified a port, replace 6379 with the actual port.
- **When connecting to a Redis Cluster instance, ensure that -c is added to the command.** Otherwise, the connection will fail.
  - Run the following command to connect to a Redis Cluster instance:

    **./redis-cli -h {dcs_instance_address} -p 6379 -a {password} -c**
  - Run the following command to connect to a single-node, master/standby, or Proxy Cluster instance:

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

  For details, see **Step 3** and **Step 4**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure (Linux)

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Install redis-cli.

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

1. Log in to the ECS.
2. Run the following command to download the source code package of your Redis client from **http://download.redis.io/releases/redis-5.0.8.tar.gz**:

   **wget http://download.redis.io/releases/redis-5.0.8.tar.gz**
3. Run the following command to decompress the **redis-3.0.7** directory from the package:

   **tar -xzf redis-5.0.8.tar.gz**
4. Run the following commands to go to the **redis-3.0.7** directory and compile the source code of your Redis client:

   **cd redis-5.0.8**

   **make**

   **cd src**

**Step 3** Access a DCS instance of a type other than Redis Cluster.

Perform the following procedure to access a DCS Redis 3.0 instance, or a single-node, master/standby, or Proxy Cluster DCS Redis 4.0 or 5.0 instance.

1. Run the following commands to access the chosen DCS Redis instance:

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

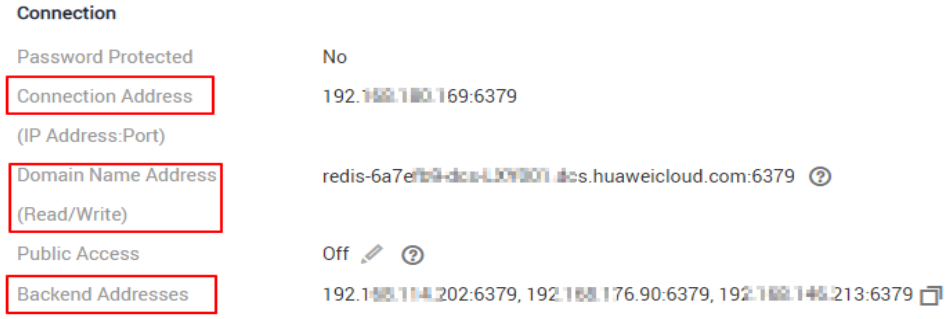
   *{dcs_instance_address}* indicates the IP address/domain name of the DCS instance and **6379** is the port used for accessing the instance. The IP address/domain name and port number are obtained in **Step 1**.

   NOTE

   For a Proxy Cluster DCS Redis 3.0 instance, you can use the **Connection Address**, **Domain Name Address**, or **Backend Addresses** for *{dcs_instance_address}*. The addresses can be obtained on the instance basic information page on the console, as shown in **Figure 4-12**.

   - **Connection Address** and **Domain Name Address** are the LB addresses. Requests are distributed across proxy nodes.
   - You can use **Backend Addresses** to directly connect to the specified proxy node.

   **Figure 4-12** Obtaining the addresses for connecting to Proxy Cluster DCS instances

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

   ```
   [root@ecs-redis redis-5.0.8]# cd src
   [root@ecs-redis src]# ./redis-cli -h redis-069949a-dcs-lxy.dcs.huaweicloud.com -p 6379
   redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379>
   ```

2. 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.

   **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>
   ```

**Step 4** Access a DCS instance of the Redis Cluster type.

Perform the following procedure to access a DCS Redis 4.0 or 5.0 instance in Redis Cluster type.

1. Run the following commands to access the chosen DCS Redis instance:

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

   *{dcs_instance_address}* indicates the IP address/domain name of the DCS Redis instance, **6379** is the port used for accessing the instance, *{password}* is

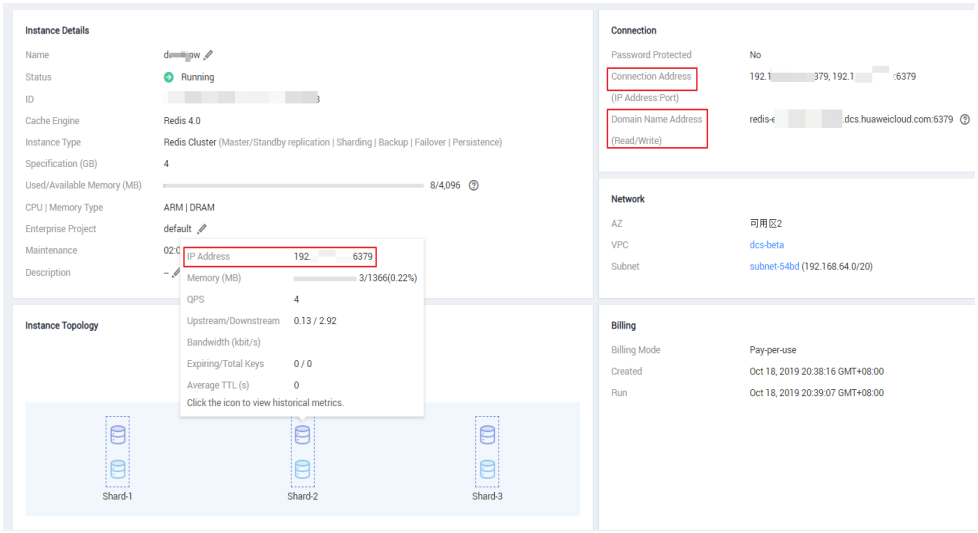
the password of the instance, and **-c** is used for accessing Redis Cluster nodes. The IP address/domain name and port number are obtained in **Step 1**.

**NOTE**

For a DCS instance of the Redis Cluster type, you can use the **Connection Address**, **Domain Name Address**, or **IP Address** in the **Instance Topology** section for *{dcs_instance_address}*. The addresses can be obtained on the instance basic information page on the console, as shown in **Figure 4-13**.

- The **Connection Address** field provides two IP addresses. You can use either of them to connect to the instance. The CRC16 of the key modulo 16384 is taken to compute what is the hash slot of a given key.
- By using the **IP Address** in the **Instance Topology** section, you can connect to the specified shard.

**Figure 4-13** Obtaining the addresses for connecting to Redis Cluster DCS instances




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

```
root@ecs-redis:~/redis-5.0.8/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-5.0.8/src# ./redis-cli -h redis-51e463c-dcs-lxy.dcs.huaweicloud.com -p
6379 -a ****** -c
redis-51e463c-dcs-lxy.dcs.huaweicloud.com:6379>
```

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

   **cluster nodes**

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

```
192.168.0.85:6379> cluster nodes
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 of the key modulo 16384 is taken to compute what is the hash slot of a given key.

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

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

**----End**

### Procedure (Windows)

**Download** the Windows Redis client installation package. Decompress the package, open the CLI tool **cmd.exe**, and go to the directory where the decompressed Redis client installation package is saved. Then, run the following command to access the DCS Redis instance:

**redis-cli -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 a DCS instance. For details on how to obtain the IP address/domain name and port, see **6.1 Viewing Instance Details**. Change the IP address/domain name and port as required.

### Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

## 4.3.2 Using Jedis to Access an Instance

Access a DCS Redis instance through Jedis on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To use Jedis to connect to a Redis Cluster instance, see **https://github.com/xetorthio/jedis#jedis-cluster**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**Step 3** Connect to the single-node, master/standby, or Proxy Cluster DCS instance by using Jedis.

1. Obtain the source code of the Jedis client from **https://github.com/xetorthio/jedis**.
2. Write code.

   Use either of the following two methods to access a DCS Redis instance through Jedis:

   – Single Jedis connection

   – Jedis pool

   Example code:

   a. Example code for a single Jedis connection

```
//Creating a connection in password mode
 String host = "192.168.0.150";
 int port = 6379;
 String pwd = "passwd";

 Jedis client = new Jedis(host, port);
 client.auth(pwd);
 client.connect();
//Run the SET command.
 String result = client.set("key-string", "Hello, Redis!");
 System.out.println( String.format("set command result:%s", result) );
//Run the GET command.
 String value = client.get("key-string");
System.out.println( String.format("get command result:%s", value) );

//Creating a connection in password-free mode
 String host = "192.168.0.150";
 int port = 6379;

 Jedis client = new Jedis(host, port);
 client.connect();
//Run the SET command.
 String result = client.set("key-string", "Hello, Redis!");
```

```
 System.out.println( String.format("set command result:%s", result) );
//Run the GET command.
 String value = client.get("key-string");
 System.out.println( String.format("get command result:%s", value) );
```

*host* indicates the example IP address/domain name of DCS instance and *port* indicates the port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address and port as required. *pwd* indicates the password used for login to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

b. Example code for a Jedis pool

```
//Generate configuration information of a Jedis pool
 String ip = "192.168.0.150";
 int port = 6379;
 String pwd = "passwd";
 GenericObjectPoolConfig config = new GenericObjectPoolConfig();
 config.setTestOnBorrow(false);
 config.setTestOnReturn(false);
 config.setMaxTotal(100);
 config.setMaxIdle(100);
 config.setMaxWaitMillis(2000);
JedisPool pool = new JedisPool(config, ip, port, 100000, pwd);//Generate a Jedis pool when the application is being initialized
//Get a Jedis connection from the Jedis pool when a service operation occurs
 Jedis client = pool.getResource();
 try {
    //Run commands
    String result = client.set("key-string", "Hello, Redis!");
    System.out.println( String.format("set command result:%s", result) );
    String value = client.get("key-string");
    System.out.println( String.format("get command result:%s", value) );
 } catch (Exception e) {
    // TODO: handle exception
 } finally {
    //Return the Jedis connection to the Jedis pool when the service operation is completed
    if (null != client) {
       pool.returnResource(client);
    }
 } // end of try block
 //Destroy the Jedis pool when the application is closed
 pool.destroy();

//Configure the connection pool in the password-free mode
 String ip = "192.168.0.150";
 int port = 6379;
 GenericObjectPoolConfig config = new GenericObjectPoolConfig();
 config.setTestOnBorrow(false);
 config.setTestOnReturn(false);
 config.setMaxTotal(100);
 config.setMaxIdle(100);
 config.setMaxWaitMillis(2000);
 JedisPool pool = new JedisPool(config, ip, port, 100000);//Generate a JedisPool when the application is being initialized
 //Get a Jedis connection from the Jedis pool when a service operation occurs
 Jedis client = pool.getResource();
 try {
    //Run commands
    String result = client.set("key-string", "Hello, Redis!");
    System.out.println( String.format("set command result:%s", result) );
    String value = client.get("key-string");
    System.out.println( String.format("get command result:%s", value) );
 } catch (Exception e) {
    // TODO: handle exception
 } finally {
    //Return the Jedis connection to the Jedis pool when the service operation is completed
    if (null != client) {
       pool.returnResource(client);
```

```
    }
} // end of try block
//Destroy the Jedis pool when the application is closed
pool.destroy();
```

*ip* indicates the IP address/domain name of DCS instance and *port* indicates the port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address and port as required. *pwd* indicates the password used for login to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

3. Compile code according to the **readme** file in the source code of the Jedis client. Run the Jedis client to access the chosen DCS Redis instance.

**----End**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.3.3 Using phpredis to Access an Instance

Access a DCS Redis instance through phpredis on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

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 **https://github.com/phpredis/phpredis/blob/develop/cluster.markdown#readme**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**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. Run the following command to download the source phpredis package:

   **wget http://pecl.php.net/get/redis-4.1.0RC3.tgz**

   The phpredis client downloaded by running the preceding command is of the latest version. To download phpredis clients of other versions, visit the Redis or PHP official website.

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

   **tar -zxvf redis-4.1.0RC3.tgz**

   **cd redis-4.1.0RC3**

3. Run the following extension command before compilation:

   **phpize**

4. Run the following command to 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. Run the following command to 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 /usr/local/php/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 = 6379;
      $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 IP address/domain name of DCS instance and **6379** is an example port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. *{password}* indicates the password used for login 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**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.3.4 Using hiredis in C++ to Access an Instance

Access a DCS Redis instance through hiredis on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To connect to a Redis Cluster instance in C++, see **https://github.com/sewenew/redis-plus-plus?_ga=2.64990636.268662337.1603553558-977760105.1588733325#redis-cluster**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**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;**

**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_ip_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 on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. *{password}* indicates the password used for login 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**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.3.5 Using redis-py to Access an Instance

Access a DCS Redis instance through redis-py on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To connect to a Redis Cluster instance in Python, see **https://github.com/Grokzen/redis-py-cluster#usage-example**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**Step 3** Install Python and redis-py.

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

   **yum install python**

2. Run the following command to download and decompress the redis-py package:

   **wget https://github.com/andymccurdy/redis-py/archive/master.zip;**

3. 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:

```
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

**Step 4** Access the DCS Redis instance by using redis-py.

In the following steps, commands are executed in CLI mode. (Alternatively, write the commands into a Python script and then execute the script.)

1. Run the **python** command to enter the CLI mode.

   You have entered CLI mode if the following command output is displayed:

```
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

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

```
r = redis.StrictRedis(host='192.168.0.171', port=6379, password='******');
```

**192.168.0.171** is an example IP address/domain name of DCS instance and **6379** is an example port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. ****** indicates the password used for login 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:

```
Python 2.6.6 (r266:84292, Aug 18 2016, 15:13:37)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-17)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>> r = redis.StrictRedis(host='192.168.0.171', port=6379, password='******');
>>>
```

**----End**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.3.6 Using Node.js to Access an Instance

Access a DCS Redis instance through Node.js on an ECS in the same VPC. For more information on how to use other Redis clients, visit **https://redis.io/clients**.

NOTE

The operations described in this section apply only to single-node, master/standby, and Proxy Cluster instances. To use Node.js to connect to a Redis Cluster instance, see **https://github.com/NodeRedis/node-redis**.

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**Step 3** Install Node.js.

**apt install nodejs-legacy**

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

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

**tar -xvf node-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 the node package manager (npm).

**apt install npm**

**Step 5** Install the Redis client ioredis.

**npm install ioredis**

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

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

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

*host* indicates the example IP address/domain name of DCS instance and *port* indicates the port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. *******indicates the password used for login to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

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

**node ioredisdemo.js**

**----End**

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

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

**Step 3** Install Node.js.

**yum install nodejs**

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

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

**tar -xvf node-v0.12.4.tar.gz;**

**cd node-v0.12.4;**

**./configure;**

**make;**

**make install;**

NOTE

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

**Step 4** Install npm.

**yum install npm**

**Step 5** Install the Redis client ioredis.

**npm install ioredis**

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

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

```
var Redis = require('ioredis');
var redis = new Redis({
  port: 6379,          // Redis port
  host: '192.168.0.196',   // Redis host
  family: 4,           // 4 (IPv4) or 6 (IPv6)
  password: '******',
  db: 0
});
redis.set('foo', 'bar');
redis.get('foo', function (err, result) {
  console.log(result);
});
```

```
// Or using a promise if the last argument isn't a function
redis.get('foo').then(function (result) {
  console.log(result);
});
// Arguments to commands are flattened, so the following are the same:
redis.sadd('set', 1, 3, 5, 7);
redis.sadd('set', [1, 3, 5, 7]);
// All arguments are passed directly to the redis server:
redis.set('key', 100, 'EX', 10);
```

*host* indicates the example IP address/domain name of DCS instance and *port* indicates the port number of DCS instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. ****** indicates the password used for login to the chosen DCS Redis instance. This password is defined during DCS Redis instance creation.

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

**node ioredisdemo.js**

**----End**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.3.7 Using C# to Access an Instance

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

## Prerequisites

- The DCS Redis instance you want to access is in the **Running** state.
- An ECS has been created to serve as your Redis client. For more information on how to create ECSs, see the *Elastic Cloud Server User Guide*.
- If the ECS runs the Linux OS, ensure that the GCC compilation environment has been installed on the ECS.

## Procedure

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

For details, see **6.1 Viewing Instance Details**.

**Step 2** Log in to the ECS.

For this example, the Windows OS is used on the ECS.

**Step 3** Install Visual Studio Community 2017 on the ECS.

**Step 4** Start Visual Studio 2017 and create a project.

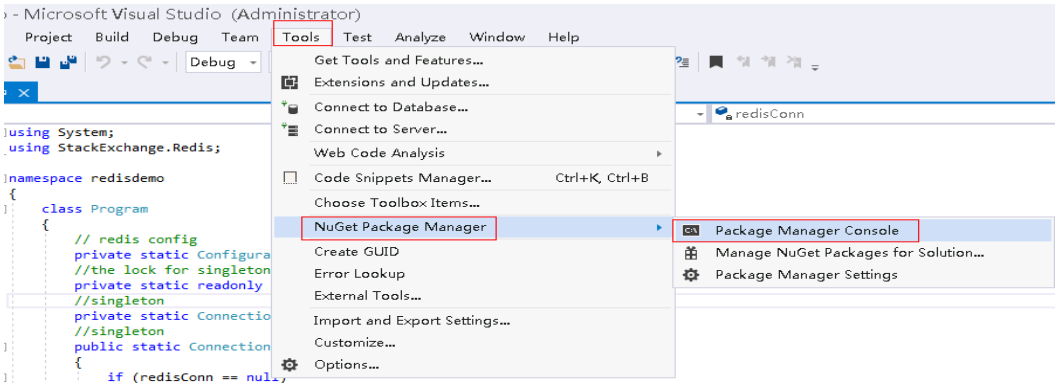
Set the project name to **redisdemo**.

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

Access the NuGet package manager console according to **Figure 4-14**, and enter **Install-Package StackExchange.Redis -*Version 1.2.6***. (The version number is optional).

**Figure 4-14** Accessing the NuGet package manager console




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

```
using System;
using StackExchange.Redis;

namespace redisdemo
{
   class Program
   {
      // redis config
      private static ConfigurationOptions connDCS =
ConfigurationOptions.Parse("10.10.38.233:6379,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();
      }
   }
}
```

*10.10.38.233:6379* contains an example IP address/domain name and port number of the DCS Redis instance. For details on how to obtain the IP address/domain name and port, see **Step 1**. Change the IP address/domain name and port as required. ******** indicates the password used for login 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**

## Video Guide

Watch the following video about accessing a DCS Redis instance.

**Accessing a DCS Instance**

# 4.4 Accessing a DCS Redis 4.0 or 5.0, or Redis 6.0 Professional Edition Instance on the Console

Access a DCS Redis instance through Web CLI. This function is supported only by DCS Redis 4.0 and 5.0, and Redis 6.0 professional edition instances, and not by DCS Redis 3.0 instances.

NOTE

Do not enter sensitive information in Web CLI to avoid disclosure.

## Prerequisites

The DCS Redis 4.0 and 5.0, and Redis 6.0 professional edition instance you want to access through Web CLI is in the **Running** state.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

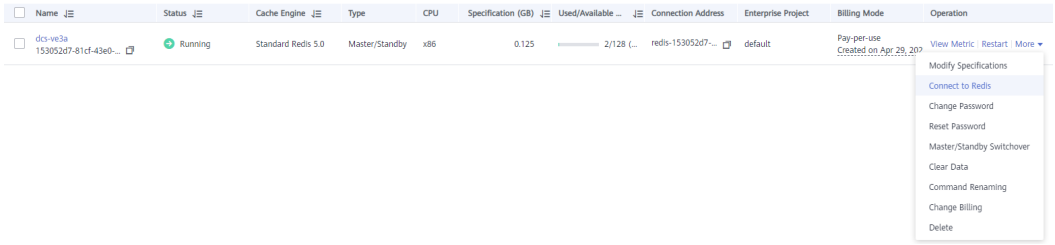
NOTE

Select the same region as your application service.

**Step 3** In the navigation pane, choose **Cache Manager**. In the **Operation** column of the instance, choose **More** > **Connect to Redis**, as shown in **Figure 4-15**.

**Figure 4-15** Accessing Web CLI




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

**----End**

# 5 Accessing a DCS Memcached Instance

## 5.1 Using telnet to Access a DCS Memcached Instance

Access a DCS Memcached instance using telnet on an ECS in the same VPC.

### Prerequisites

- The DCS Memcached instance you want to access is in the **Running** state.
- An ECS has been created on which the client has been installed. For details on how to create ECSs, see the *Elastic Cloud Server User Guide*.

  NOTE

  1. An ECS can communicate with a DCS instance that belongs to the same VPC and is configured with the same security group.
  2. If the ECS and DCS instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**
  3. If different security groups have been configured for the ECS and DCS instance, set security group rules to achieve network connectivity between the ECS and DCS instance. For details, see **How Do I Configure a Security Group?**

- All annotations in example code have been deleted.
- All command lines and code blocks are UTF-8 encoded. Using another encoding scheme will cause compilation problems or even command failures.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** On the **Cache Manager** page, click the name of the DCS Memcached instance you want to access. Obtain the IP address or domain name and port number of the instance.

**Step 5** Access the chosen DCS Memcached instance.

1. Log in to the ECS.
2. Run the following command to check whether telnet is installed on the ECS:

   ***which telnet***

   If the directory in which the telnet is installed is displayed, telnet has been installed on the ECS. If the client installation directory is not displayed, install the telnet manually.

   NOTE

   - If telnet has not been installed in Linux, run the **yum -y install telnet** command to install it.
   - In the Windows OS, choose **Start** > **Control Panel** > **Programs** > **Programs and Features** > **Turn Windows features on or off**, and enable telnet.

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

   ***telnet {ip or domain name} {port}***

   In this command: ***{ip address or domain name}*** indicates the IP address or domain name of the DCS Memcached instance. ***{port}*** indicates the port number of the DCS Memcached instance. Both the IP address or domain name and the port number are obtained in **Step 4**.

   When you have successfully accessed the chosen DCS Memcached instance, information similar to the following is displayed:

   ```
   Trying XXX.XXX.XXX.XXX...
   Connected to XXX.XXX.XXX.XXX.
   Escape character is '^]'.
   ```

   NOTE

   - If **Password-protected** is not enabled for the instance, run the following commands directly after the instance is accessed successfully.
   - If **Password-protected** is enabled for the instance, attempts to perform operations on the instance will result in the message "ERROR authentication required", indicating that you do not have the required permissions. In this case, enter **auth *username@password*** to authenticate first. *username* and *password* are that used for accessing the DCS Memcached instance.

   Example commands for using the DCS Memcached instance (lines in red are the commands and the other lines are the command output):

   ```
   set hello 0 0 6
   world!
   STORED
   get hello
   VALUE hello 0 6
   world!
   END
   ```

**----End**

# 5.2 Using Java to Access an Instance

Access a DCS Memcached instance using a Java client on an ECS in the same VPC.

## Prerequisites

- The DCS Memcached instance you want to access is in the **Running** state.
- An ECS has been created on which the client has been installed. For details on how to create ECSs, see the *Elastic Cloud Server User Guide*.

  NOTE

  An ECS can communicate with a DCS instance that belongs to the same VPC and is configured with the same security group.

  If the ECS and DCS instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**

  If different security groups have been configured for the ECS and DCS instance, set security group rules to achieve network connectivity between the ECS and DCS instance. For details, see **How Do I Configure a Security Group?**
- The Java development kit (JDK) and common integrated development environments (IDEs) such as Eclipse have been installed on the ECS.
- You have obtained the **spymemcached-x.y.z.jar** dependency package.

  NOTE

  *x.y.z* indicates the version of the dependency package. The latest version is recommended.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** On the **Cache Manager** page, click the name of the DCS Memcached instance you want to access. Obtain the IP address or domain name and port number of the instance.

**Step 5** Upload the obtained **spymemcached-x.y.z.jar** dependency package to the created ECS.

**Step 6** Log in to the ECS.

**Step 7** Create a Java project on Eclipse and import the **spymemcached-x.y.z.jar** dependency package. The project name is customizable.

**Step 8** Create a **ConnectMemcached1** class, copy the following Java code to the class, and modify the code.

- Example code for the password mode

  Change **ip address or domain name:port** to the IP address and port number obtained in **Step 4**. Set userName and password respectively to the username and password of the Memcached instance.

```
//Connect to the encrypted Memcached code using Java.
import java.io.IOException;
import java.util.concurrent.ExecutionException;

import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.ConnectionFactoryBuilder.Protocol;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.auth.AuthDescriptor;
import net.spy.memcached.auth.PlainCallbackHandler;
import net.spy.memcached.internal.OperationFuture;

public class ConnectMemcached1
{
    public static void main(String[] args)
    {
        final String connectionaddress = "ip or domain name:port";
        final String username = "userName";//Indicates the username.
        final String password = "password";//Indicates the password.
        MemcachedClient client = null;
        try
        {
            AuthDescriptor authDescriptor =
                new AuthDescriptor(new String[] {"PLAIN"}, new PlainCallbackHandler(username,
                    password));
            client = new MemcachedClient(
                    new ConnectionFactoryBuilder().setProtocol(Protocol.BINARY)
                        .setAuthDescriptor(authDescriptor)
                        .build(),
                    AddrUtil.getAddresses(connectionaddress));
            String key = "memcached";//Stores data with the key being memcached in Memcached.
            String value = "Hello World";//The value is Hello World.
            int expireTime = 5; //Specifies the expiration time, measured in seconds. The countdown
starts from the moment data is written. After the expireTime elapses, the data expires and can no
longer be read.
            doExcute(client, key, value, expireTime);//Executes the operation.
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *Method of writing data to Memcached
     */
    private static void doExcute(MemcachedClient client, String key, String value, int expireTime)
    {
        try
        {
            OperationFuture<Boolean> future = client.set(key, expireTime, value);
            future.get();//spymemcached set () is asynchronous. future.get () waits until the cache.set
() operation is completed, or does not need to wait. You can select based on actual requirements.
            System.out.println("The Set operation succeeded.");
            System.out.println("Get operation:" + client.get(key));
            Thread.sleep(6000);//Waits for 6000 ms, that is, 6s. Then the data expires and can no longer
be read.
            System.out.println("Perform the Get operation 6s later:" + client.get(key));

        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        catch (ExecutionException e)
        {
            e.printStackTrace();
        }
        if (client != null)
```

```
        {
            client.shutdown();
        }
    }
}
```

- Example code for the password-free mode

  Change **ip address or domain name:port** to the IP address and port number obtained in **Step 4**.

```
//Connect to the password-free Memcached code using Java.
import java.io.IOException;
import java.util.concurrent.ExecutionException;

import net.spy.memcached.AddrUtil;
import net.spy.memcached.BinaryConnectionFactory;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.internal.OperationFuture;

public class ConnectMemcached
{
    public static void main(String[] args)
    {
        final String connectionaddress = "ip or domain name:port";
        MemcachedClient client = null;
        try
        {
            client = new MemcachedClient(new BinaryConnectionFactory(),
AddrUtil.getAddresses(connectionaddress));
            String key = "memcached";//Stores data with the key being memcached in Memcached.
            String value = "Hello World";//The value is Hello World.
            int expireTime = 5; //Specifies the expiration time, measured in seconds. The countdown
starts from the moment data is written. After the expireTime elapses, the data expires and can no
longer be read.
            doExcute(client, key, value, expireTime);//Executes the operation.
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *Method of writing data to Memcached
     */
    private static void doExcute(MemcachedClient client, String key, String value, int expireTime)
    {
        try
        {
            OperationFuture<Boolean> future = client.set(key, expireTime, value);
            future.get();//spymemcached set () is asynchronous. future.get () waits until the cache.set
() operation is completed, or does not need to wait. You can select based on actual requirements.
            System.out.println("The Set operation succeeded.");
            System.out.println("Get operation:" + client.get(key));
            Thread.sleep(6000);//Waits for 6000 ms, that is, 6s. Then the data expires and can no longer
be read.
            System.out.println("Perform the Get operation 6s later:" + client.get(key));

        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
        catch (ExecutionException e)
        {
            e.printStackTrace();
        }
        if (client != null)
        {
            client.shutdown();
```

```
    }
  }
}
```

**Step 9** Run the **main** method. The following result is displayed in the **Console** window of Eclipse:

```
The Set operation succeeded.
Get operation: Hello World
Perform the Get operation 6s later: null
```

**----End**

# 5.3 Using Python to Access an Instance

Access a DCS Memcached instance using Python on an ECS in the same VPC.

## Prerequisites

- The DCS Memcached instance you want to access is in the **Running** state.
- Log in to the ECS. For details on how to create ECSs, see the *Elastic Cloud Server User Guide*.

  NOTE

  An ECS can communicate with a DCS instance that belongs to the same VPC and is configured with the same security group.

  If the ECS and DCS instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**

  If different security groups have been configured for the ECS and DCS instance, set security group rules to achieve network connectivity between the ECS and DCS instance. For details, see **How Do I Configure a Security Group?**
- Python has been installed on the ECS. The recommended version is 2.7.6 or later.
- You have obtained the **python-binary-memcached-x.y.z.zip** dependency package.

  NOTE

  *x.y.z* indicates the version of the dependency package. The latest version is recommended.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** On the **Cache Manager** page, click the name of the DCS Memcached instance you want to access. Obtain the IP address or domain name and port number of the instance.

**Step 5** Upload the obtained dependency package (for example, the **python-binary-memcached-x.y.z.zip** package) to the created ECS.

**Step 6** Log in to the ECS.

**Step 7** Run the following commands to install the dependency package:

**unzip -xzvf python-binary-memcached-x.y.z.zip**

**cd python-binary-memcached-x.y.z**

**python setup.py install**

NOTE

If an error is reported during the installation, use the **apt** or **yum** installation method. For example, to install the dependency package by using the **apt** method, run the following commands:

**apt install python-pip;**

**pip install python-binary-memcached;**

**Step 8** Create a Python file named **dcs_test.py**, copy the following Python code to the file, and modify the code.

- Example code for the password mode

  Change **ip address or domain name:port** to the IP address or domain name and port number obtained in **Step 4**. Set userName and password respectively to the username and password of the Memcached instance.

```
import bmemcached
client = bmemcached.Client(('ip or domain name:port'), 'userName', 'password')
print "set('key', 'hello world!')"
print client.set('key', 'hello world!')
print "get('key')"
print client.get('key')
```

- Example code for the password-free mode

  Change **ip address or domain name:port** to the IP address and port number obtained in **Step 4**.

```
import bmemcached
client = bmemcached.Client('ip or domain name:port')
print "set('key', 'hello world!')"
print client.set('key', 'hello world!')
print "get('key')"
print client.get('key')
```

**Step 9** Run the **dcs_test.py** file. The following result is displayed.

```
# python test.py
set('key', 'hello world!')
True
get('key')
hello world!
```

**----End**

# 5.4 Using C++ to Access an Instance

Access a DCS Memcached instance using a C++ client on an ECS in the same VPC.

## Prerequisites

- The DCS Memcached instance you want to access is in the **Running** state.
- Log in to the ECS. For details on how to create ECSs, see the *Elastic Cloud Server User Guide*.

  NOTE

  An ECS can communicate with a DCS instance that belongs to the same VPC and is configured with the same security group.

  If the ECS and DCS instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**

  If different security groups have been configured for the ECS and DCS instance, set security group rules to achieve network connectivity between the ECS and DCS instance. For details, see **How Do I Configure a Security Group?**
- GCC has been installed on the ECS. The recommended version is 4.8.4 or later.
- You have obtained the **libmemcached-x.y.z.tar.gz** dependency package.

  NOTE

  *x.y.z* indicates the version of the dependency package. The latest version is recommended.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** On the **Cache Manager** page, click the name of the DCS Memcached instance you want to access. Obtain the IP address or domain name and port number of the instance.

**Step 5** Upload the obtained **libmemcached-x.y.z.tar.gz** dependency package to the created ECS.

**Step 6** Log in to the ECS.

**Step 7** Install related SASL dependency packages.

For OSs of Debian series: **apt install libsasl2-dev cloog.ppl**

For OSs of Red Hat series: **yum install cyrus-sasl***

**Step 8** Run the following commands to install the dependency package:

***tar -xzvf libmemcached-x.y.z.tar.gz***

***cd libmemcached-x.y.z***

***./configure --enable-sasl***

***make***

***make install***

**Step 9** Create a file named **build.sh** and copy the following code to the file.

```
g++ -o dcs_sample dcs_sample.cpp -lmemcached -std=c++0x -lpthread -lsasl2
```

NOTE

If the **libmemcached.so.11** file cannot be found during compilation, run the **find** command to find the file and copy the file to the **/usr/lib** directory.

**Step 10** Create a file named **dcs_sample.cpp**, copy the following C++ code to the file, and modify the code.

- Example code for the password mode

  Change *ip or domain name* to the IP address or domain name and port number obtained in **Step 4**. Set userName and password respectively to the username and password of the Memcached instance.

```
#include <iostream>
#include <string>
#include <libmemcached/memcached.h>
using namespace std;

#define IP  "ip or domain name"
#define PORT  port
#define USERNAME "userName"
#define PASSWORD "password"
memcached_return rusult;

memcached_st * init()
{
    memcached_st *memcached = NULL;
    memcached_server_st *cache;
    memcached = memcached_create(NULL);
    cache = memcached_server_list_append(NULL, IP, PORT, &rusult);

    sasl_client_init(NULL);
        memcached_set_sasl_auth_data(memcached, USERNAME, PASSWORD);
        memcached_behavior_set(memcached,MEMCACHED_BEHAVIOR_BINARY_PROTOCOL,1);
        memcached_server_push(memcached,cache);
        memcached_server_list_free(cache);
          return memcached;
}

int main(int argc, char *argv[])
{
        memcached_st *memcached=init();
    string key = "memcached";
    string value = "hello world!";
    size_t value_length = value.length();
    int expire_time = 0;
    uint32_t flag = 0;

    rusult =
memcached_set(memcached,key.c_str(),key.length(),value.c_str(),value.length(),expire_time,flag);
    if (rusult != MEMCACHED_SUCCESS){
      cout <<"set data failed: " << rusult << endl;
      return -1;
    }
    cout << "set succesed, key: " << key << ", value: " << value << endl;
    cout << "get key:" << key << endl;
    char* result = memcached_get(memcached,key.c_str(),key.length(),&value_length,&flag,&rusult);
    cout << "value:" << result << endl;

    memcached_free(memcached);
    return 0;
}
```

- Example code for the password-free mode

  Change *ip or domain name* to the IP address or domain name and port number obtained in **Step 4**.

```
#include <iostream>
#include <string>
#include <libmemcached/memcached.h>
using namespace std;

#define IP  "ip or domain name"
#define PORT  port
memcached_return rusult;

memcached_st * init()
{
    memcached_st *memcached = NULL;
    memcached_server_st *cache;
    memcached = memcached_create(NULL);
    cache = memcached_server_list_append(NULL, IP, PORT, &rusult);
        memcached_server_push(memcached,cache);
    memcached_server_list_free(cache);
        return memcached;
}

int main(int argc, char *argv[])
{
        memcached_st *memcached=init();
    string key = "memcached";
    string value = "hello world!";
    size_t value_length = value.length();
    int expire_time = 0;
    uint32_t flag = 0;

    rusult =
memcached_set(memcached,key.c_str(),key.length(),value.c_str(),value.length(),expire_time,flag);
    if (rusult != MEMCACHED_SUCCESS){
      cout <<"set data failed: " << rusult << endl;
      return -1;
    }
    cout << "set succesed, key: " << key << " ,value: " << value << endl;
    cout << "get key:" << key << endl;
    char* result = memcached_get(memcached,key.c_str(),key.length(),&value_length,&flag,&rusult);
    cout << "value:" << result << endl;

    memcached_free(memcached);
    return 0;
}
```

**Step 11** Run the following commands to compile the source code:

chmod 700 build.sh

./build.sh

The **dcs_sample** binary file is generated.

**Step 12** Run the following command to access the chosen DCS Memcached instance:

./dcs_sample

```
set succesed, key: memcached ,value: hello world!
get key:memcached
value:hello world!
```

**----End**

# 5.5 Using PHP to Access an Instance

Access a DCS Memcached instance in PHP on an ECS in the same VPC.

## Prerequisites

- The DCS Memcached instance you want to access is in the **Running** state.
- Log in to the ECS. For details on how to create ECSs, see the *Elastic Cloud Server User Guide*.

  NOTE

  An ECS can communicate with a DCS instance that belongs to the same VPC and is configured with the same security group.

  If the ECS and DCS instance are in different VPCs, establish a VPC peering connection to achieve network connectivity between the ECS and DCS instance. For details, see **Does DCS Support Cross-VPC Access?**

  If different security groups have been configured for the ECS and DCS instance, set security group rules to achieve network connectivity between the ECS and DCS instance. For details, see **How Do I Configure a Security Group?**

## For OSs of Red Hat series:

The following uses CentOS 7.0 as an example to describe how to install a PHP client and use it to access a DCS Memcached instance. The procedure is also applicable to a PHP client running the Red Hat or Fedora OS.

**Step 1** Install GCC-C++ and Make compilation components.

**yum install gcc-c++ make**

**Step 2** Install related SASL packages.

**yum install cyrus-sasl***

**Step 3** Install the libMemcached library.

Installing the libMemcached library requires SASL authentication parameters. Therefore, you cannot install the library by running the **yum** command.

**wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz**

**tar -xvf libmemcached-1.0.18.tar.gz**

**cd libmemcached-1.0.18**

**./configure --prefix=/usr/local/libmemcached --enable-sasl**

**make && make install**

NOTE

Before installing the libMemcached library, install GCC-C++ and SASL components. Otherwise, an error will be reported during compilation. After you resolve the error, run the **make clean** command and then run the **make** command again.

**Step 4** Install the PHP environment.

**yum install php-devel php-common php-cli**

**NOTICE**

PHP 7.x does not support SASL authentication. Use PHP 5.6. If the yum php version is not 5.6, download one from the Internet.

**Step 5** Install the Memcached client.

Note that you must add a parameter used to enable SASL when running the **configure** command.

**wget http://pecl.php.net/get/memcached-2.1.0.tgz**

**tar zxvf memcached-2.1.0.tgz**

**cd memcached-2.1.0**

**phpize**

**./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl**

**make && make install**

**Step 6** Modify the **php.ini** file.

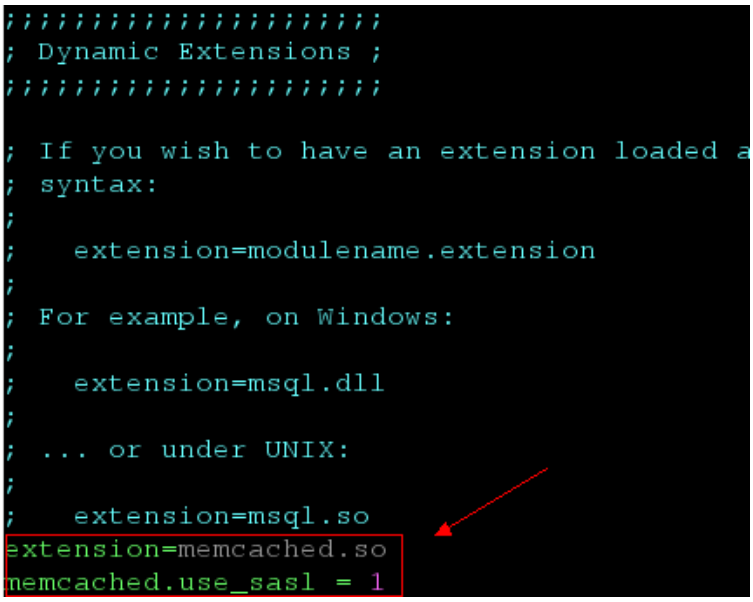
Run the **find** or **locate** command to find the **php.ini** file.

**find / -name php.ini**

Add the following two lines to the **php.ini** file:

```
extension=memcached.so
memcached.use_sasl = 1
```

**Figure 5-1** Modifying the **php.ini** file



**Step 7** Access a DCS Memcached instance.

Create a **memcached.php** file and add the following content to the file:

```
<?php
    $connect = new Memcached; //Declares a Memcached connection.
    $connect->setOption(Memcached::OPT_COMPRESSION, false); //Disables compression.
    $connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true); //Uses the binary protocol.
    $connect->setOption(Memcached::OPT_TCP_NODELAY, true); //Disables the TCP network delay policy.
    $connect->addServer('{memcached_instance_ip}', 11211); //Specifies the instance IP address and port number.
    $connect->setSaslAuthData('{username}', '{password}'); //If password-free access is enabled for the instance, delete or comment out this line.
    $connect->set("DCS", "Come on!");
    echo 'DCS: ',$connect->get("DCS");
    echo "\n";
    $connect->quit();
?>
```

Save and run the **memcached.php** file. The following result is displayed.

```
[root@testphpmemcached ~]# php memcached.php
    DCS: Come on!
[root@testphpmemcached ~]#
```

**----End**

## For OSs of Debian series:

The following uses the Ubuntu OS as an example to describe how to install a PHP client and use it to access a DCS Memcached instance.

**Step 1** Install GCC and Make compilation components.

**apt install gcc make**

**Step 2** Install the PHP environment.

PHP 5.x is recommended for better compatibility with SASL authentication.

Run the following commands to add the image source of PHP of an earlier version, and then install the **php.56** and **php.5.6-dev** packages:

**apt-get install -y language-pack-en-base;**

**LC_ALL=en_US.UTF-8;**

**add-apt-repository ppa:ondrej/php;**

**apt-get update;**

**apt-get install php5.6 php5.6-dev;**

After the installation is complete, run the **php -version** command to check the PHP version. If the following result is displayed, the PHP version is 5.6, indicating that PHP 5.6 is successfully installed.

```
root@dcs-nodelete:/etc/apt# php -version
PHP 5.6.36-1+ubuntu16.04.1+deb.sury.org+1 (cli)
Copyright (c) 1997-2016 The PHP Group
```

NOTE

To uninstall PHP, run the following commands:

**apt install aptitude -y**

**aptitude purge `dpkg -l | grep php| awk '{print $2}' |tr "\n" " "`**

**Step 3** Install the SASL component.

**apt install libsasl2-dev cloog.ppl**

**Step 4** Install the libMemcached library.

**wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz**

**tar -xvf libmemcached-1.0.18.tar.gz**

**cd libmemcached-1.0.18**

**./configure --prefix=/usr/local/libmemcached**

**make && make install**

NOTE

Before installing the libMemcached library, install GCC-C++ and SASL components. Otherwise, an error will be reported during compilation. After you resolve the error, run the **make clean** command and then run the **make** command again.

**Step 5** Install the Memcached client.

Install the zlib component.

**apt install zlib1g.dev**

Note that you must add a parameter used to enable SASL when running the **configure** command.

**wget http://pecl.php.net/get/memcached-2.2.0.tgz**;

**tar zxvf memcached-2.2.0.tgz**;

**cd memcached-2.2.0**;

**phpize5.6**;

**./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl**;

**make && make install**;

**Step 6** Modify the **pdo.ini** file.

Run the following command to find the **pdo.ini** file:

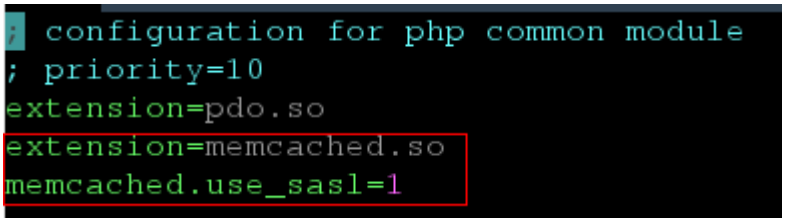
**find / -name pdo.ini**

By default, the **pdo.ini** file is stored in the **/etc/php/5.6/mods-available** directory. Add the following two lines to the **php.ini** file:

```
extension=memcached.so
memcached.use_sasl = 1
```

**Figure 5-2** Modifying the **pdo.ini** file




**Step 7** Access a DCS Memcached instance.

Create a **memcached.php** file and add the following content to the file:

```
<?php
    $connect = new Memcached; //Declares a Memcached connection.
    $connect->setOption(Memcached::OPT_COMPRESSION, false); //Disables compression.
    $connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true); //Uses the binary protocol.
    $connect->setOption(Memcached::OPT_TCP_NODELAY, true); //Disables the TCP network delay policy.
    $connect->addServer('{memcached_instance_ip}', 11211); //Specifies the instance IP address and port number.
    $connect->setSaslAuthData('{username}', '{password}'); //If password-free access is enabled for the instance, delete or comment out this line.
    $connect->set("DCS", "Come on!");
    echo 'DCS: ',$connect->get("DCS");
    echo "\n";
    $connect->quit();
?>
```

Save and run the **memcached.php** file. The following result is displayed.

```
[root@dcs-nodelete ~]# php memcached.php
    DCS: Come on!
[root@dcs-nodelete ~]#
```

**----End**

# 6 Operating DCS Instances

## 6.1 Viewing Instance Details

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

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

**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, ID, IP address, AZ, status, instance type, cache engine, CPU, enterprise project, billing mode, and tags.

  For example, to filter DCS instances by cache engine or cache engine version, click the search box, choose **Cache Engine**, and then choose **Redis**, **Redis 3.0**, **Redis 4.0**, **Redis 5.0**, **Redis 6.0**, or **Memcached**.

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

**Step 5** On the DCS instance list, click the name of a DCS instance to display more details about it. **Table 6-1** describes the parameters.

**Table 6-1** Parameters on the Basic Information page of a DCS instance

| Section | Parameter | Description |
| --- | --- | --- |
| Instance Details | Name | Name of the chosen instance. To modify the instance name, click the icon. |
| | Status | State of the chosen instance. |
| | ID | ID of the chosen instance. |
| | Cache Engine | Cache engine used by the DCS instance, which can be Redis or Memcached. If the cache engine is Redis, it is followed by the version number, for example, Redis 3.0. |
| | Instance Type | Type of the selected instance. Currently, supported types include single-node, master/standby, Proxy Cluster, read/write splitting, and Redis Cluster. |
| | Cache Size | Specification of the chosen instance. |
| | Used/ Available Memory (MB) | The used memory space and maximum available memory space of the chosen instance.<br>The used memory space includes:<br>• Size of data stored on the DCS instance<br>• Size of Redis-server buffers (including client buffer and repl-backlog) and internal data structures |
| | CPU | CPU architecture of the chosen instance. This parameter is displayed only for DCS Redis instances. |
| | Enterprise Project | Enterprise project to which the new instance belongs. Click to view the enterprise project of the instance. |
| | Maintenance | Time range for any scheduled maintenance activities on cache nodes of this DCS instance. To modify the time window, click the icon. |
| | Description | Description of the chosen DCS instance. To modify the description, click the icon. |
| Connection | Password Protected | Password-protected or password-free access. |

| Section | Parameter | Description |
| --- | --- | --- |
| | Connection Address | Domain name and port number of the instance.<br>For a master/standby Redis 4.0 or 5.0, or Redis 6.0 professional edition instance, this address indicates the domain name and port number of the master node. **Read-only Address** is the domain name and port number 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. |
| | IP Address | IP address and port number of the instance. The domain name connection address is recommended. |
| | Public Access | An indicator of whether public access is enabled. **Public access is only supported for Redis 3.0, and not for Redis 4.0, Redis 5.0, Redis 6.0 professional edition, and Memcached**. |
| | Public Access Address | Elastic IP address bound to the instance for public access. This parameter is displayed only when **Public Access** is enabled.<br>**NOTE**<br>Click **Download Certificate for Public Access** to download a certificate, which can be used to verify the certificate of the DCS instance when you access the instance. |
| Network | AZ | Availability zone in which the cache node running the selected DCS instance resides. |
| | VPC | VPC in which the chosen instance resides. |
| | Subnet | Subnet in which the chosen instance resides. |
| | Security Group | Security group that controls access to the chosen instance. To modify the security group, click the icon. This parameter is displayed only for DCS Redis 3.0 and Memcached instances. DCS for Redis 4.0 and 5.0, and Redis 6.0 professional edition are based on VPC Endpoint and do not support security groups. |
| Instance Topology | - | Hover over a node to view its metrics, or click the icon of a node to view its historical metrics.<br>Topologies are supported only for master/standby, Proxy Cluster, and Redis Cluster instances. |
| 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**

# 6.2 Modifying Specifications

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

For example, you can change a 2 GB single-node DCS Redis 3.0 instance to a master/standby instance with another cache size.

NOTE

- **Modify instance specifications during off-peak hours.**
- If your DCS instances are too old to support scaling, contact technical support to upgrade the instances.
- Redis 6.0 (professional edition) does not support specification modification.

## Change of the Instance Type

- **Supported instance type changes:**
  - From single-node to master/standby: Supported by Redis 3.0 and Memcached, and not by Redis 4.0 and 5.0.
  - From master/standby to Proxy Cluster: Supported by Redis 3.0, and not by Redis 4.0 and 5.0.

    If a master/standby DCS Redis 3.0 instance has multiple databases, or if its data is not stored on DB0, the instance cannot be changed to the Proxy Cluster type. A master/standby instance can be changed to the Proxy Cluster type only if its data is stored and only stored on DB0.
  - From cluster types to other types: Not supported.
- **Impact of instance type changes:**
  - From single-node to master/standby:

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

    The instance cannot be connected and remains read-only for 5 to 30 minutes.

## Scale-In and Scale-Out

- **The following table lists scaling options supported by different DCS instances.**

  **Table 6-2** Scaling options supported by different DCS instances

| Cache Engine | Single-Node | Master/Standby | Cluster |
| --- | --- | --- | --- |
| Redis 3.0 | Increase and decrease | Increase and decrease | Increase |

| Cache Engine | Single-Node | Master/Standby | Cluster |
| --- | --- | --- | --- |
| Redis 4.0 | Increase and decrease | Increase and decrease | Increase |
| Redis 5.0 | Increase and decrease | Increase and decrease | Increase |
| Memcached | Increase and decrease | Increase and decrease | N/A |

- **Impact of scaling:**
  - Single-node and master/standby:

    For a DCS Redis 3.0, 4.0, or 5.0 instance, connections are interrupted for several seconds and the instance remains read-only for about 1 minute.

    During scaling, only the memory of the instance is expanded. The CPU processing capability is not improved.

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

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

    Backup records created before scaling cannot be restored.
  - Redis Cluster:

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

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

**Step 4** Choose **More** > **Modify Specifications** in the row containing the DCS instance.

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

NOTE

If new replicas are added to a Redis Cluster instance, the **Added Replicas** field will be displayed on the page. Modifying the instance specification will cause the replica specification to be modified, which results in a change in price.

**Step 6** Click **Next**, confirm the details, and click **Submit**.

You can go to **Background Tasks** page to view the modification status. For more information, see **7.5 Viewing Background Tasks**.

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

NOTE

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

**----End**

# 6.3 Starting an Instance

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

When a cluster instance is started, status and data are synchronized between the nodes of the instance. If a large amount of data is continuously written into the instance before the synchronization is complete, the synchronization will be prolonged and the instance remains in the **Starting** state. After the synchronization is complete, the instance enters the **Running** state.

NOTE

This function is supported only by old DCS Redis instances in the **Stopped** state. New instance cannot be started or stopped.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Select the instance for which you want to start and click **Start** above the DCS instance list.

**Step 5** In the displayed dialog box, click **Yes**.

- It takes 1 to 30 minutes to start DCS instances.
- After DCS instances are started, their statuses change from **Stopped** to **Running**.

NOTE

To start a single instance, click **Start** in the **Operation** column in the same row as the instance.

**----End**

# 6.4 Restarting an Instance

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

WARNING

- After a single-node DCS instance is restarted, data will be deleted from the instance.
- While a DCS instance is restarting, it cannot be read from or written to.
- An attempt to restart a DCS instance while it is being backed up may result in a failure.

## Prerequisites

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

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

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

**Step 5** Click **Restart** above the DCS instance list.

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

It takes 1 to 30 minutes to restart DCS instances. After DCS instances are restarted, their status changes to **Running**.

NOTE

- By default, only the instance process will restart. However, selecting **Forced Restart** will restart the VM on which the chosen DCS instance runs.
- To restart a single instance, you can also click **Restart** in the same row as the instance.

**----End**

# 6.5 Deleting an Instance

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

NOTICE

- After a DCS instance is deleted, the instance data will be deleted without backup. In addition, any backup data of the instance will be deleted. Therefore, download the backup files of the instance for permanent storage before deleting the instance.
- If the instance is in cluster mode, all cluster nodes will be deleted.

## Prerequisites

- The DCS instances you want to delete have been created.
- The DCS instances you want to delete are in the **Running**, **Faulty**, or **Stopped** state.
- To achieve fine-grained management of your HUAWEI CLOUD resources, create IAM user groups and users and grant specified permissions to the users. For details, see **2 Permissions Management**.

## Procedure

Deleting DCS Instances

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

It takes 1 to 30 minutes to delete DCS instances.

NOTE

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

**----End**

**Deleting Instance Creation Tasks That Have Failed to Run**

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

If there are DCS instances that have failed to be created, **Instance Creation Failures** 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 same row as the task.

**----End**

# 6.6 Performing a Master/Standby Switchover

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

Only master/standby instances support a master/standby switchover.

To perform a manual switchover for a Proxy Cluster or Redis Cluster DCS Redis 4.0 or 5.0 instance, go to the **Shards and Replicas** page of the instance. For details, see **7.8 Managing Shards and Replicas**.

NOTICE

- During a master/standby node 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.

### Prerequisites

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

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** In the **Operation** column of the instance, choose **More** > **Master/Standby Switchover**.

**----End**

## 6.7 Clearing DCS Instance Data

On the DCS console, you can clear data only for DCS Redis 4.0 and 5.0, and Redis 6.0 professional edition instances. Clearing instance data cannot be undone and cleared data cannot be recovered. Exercise caution when performing this operation.

### Prerequisites

The DCS Redis 4.0 or 5.0, or Redis 6.0 professional edition instance is in the **Running** state.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

**Step 4** Select one or more DCS2.0 instances to clear.

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

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

**----End**

# 6.8 Exporting Instance List

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

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

**Step 4** Filter DCS instances to find the desired DCS instance.

**Step 5** Click above the instance list.

Click the export result displayed in the lower left corner of the page. **Figure 6-1** shows an example result.

**Figure 6-1** Exported DCS instance list

| Name | ID | Status | AZ | Cache Eng | Instance | Specifica | Used/Avai | Connecti | Created | Billing M | VPC | VPC ID | Enterprise Project |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dcs-trpt | 5e4f4c58- | Running | AZ1 | Redis 5.0 | Single-no | 0.125 | 0/128 (0% | 198.19.32 | May 24, 2 | Free | null | null | default |
| dcs-APITe | 693491b0- | Running | | Redis 3.0 | Master/St | 2 | 2/1,536 ( | 172.16.14 | May 06, 2 | Yearly/M | null | 52267da0- | default |

**----End**

# 6.9 Renaming Commands

After creating a DCS Redis 4.0 or 5.0, or Redis 6.0 professional edition instance, you can rename the following critical commands: Currently, you can only rename the **COMMAND**, **KEYS**, **FLUSHDB**, **FLUSHALL**, and **HGETALL** commands.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner and select a region and a project.

**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**.

NOTE

- You can rename multiple commands at a time.
- The new command names will take effect only after you restart the instance. Remember the new command names because they will not be displayed on the console for security purposes.

**----End**

# 7 Managing DCS Instances

## 7.1 Configuration Notice

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

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

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

**NOTICE**

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

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

# 7.2 Modifying Configuration Parameters

## 7.2.1 Modifying Configuration Parameters of an Instance

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

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

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

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

**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, click the **Parameters** tab.

**Step 6** On the **Parameters** tab page, click **Modify**.

**Step 7** Modify parameters based on your requirements.

**Table 7-1** and **Table 7-2** describe the parameters. In most cases, default values are retained.

**Table 7-1** DCS Redis instance configuration parameters

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| timeout | The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of **0** means that this function is disabled. | 0–7200 seconds | 0 |

| Parameter | Description | 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.<br>There are three settings:<br>no: fsync() is never called. The OS will flush data when it is ready. This mode offers the highest performance.<br>always: fsync() is called after every write to the AOF. This mode is very slow, but also very safe.<br>everysec: fsync() is called once per second. This mode provides a compromise between safety and performance. | • no<br>• always<br>• everysec | everysec |
| 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. Options:<br>**yes**: Logs are enabled, that is, persistence is enabled.<br>**no**: Logs are disabled, that is, persistence is disabled. | • yes<br>• no | yes |
| client-output-buffer-limit-slave-soft-seconds | Number of seconds that the output buffer remains above **client-output-buffer-slave-soft-limit** before the client is disconnected. | 0–60 | 60 |

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| client-output-buffer-slave-hard-limit | Hard limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the hard limit, the client is immediately disconnected. | 0–17,179,869,184 | 1,717,986,918 |
| client-output-buffer-slave-soft-limit | Soft limit (in bytes) on the output buffer of replica clients. Once the output buffer exceeds the soft limit and continuously remains above the limit for the time specified by the **client-output-buffer-limit-slave-soft-seconds** parameter, the client is disconnected. | 0–17,179,869,184 | 1,717,986,918 |
| maxmemory-policy | The policy applied when the maxmemory limit is reached.<br>For more information about this parameter, see **https://redis.io/topics/lru-cache**. | • volatile-lru<br>• allkeys-lru<br>• volatile-random<br>• allkeys-random<br>• volatile-ttl<br>• noeviction | volatile-lru<br>**NOTE**<br>If the DCS Redis instance is created before July 2020 and this parameter has not been modified, the default value is **noeviction**. If the instance is created after July 2020, the default value is **volatile-lru**. |

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| lua-time-limit | Maximum time allowed for executing a Lua script (in milliseconds). | 100–5,000 | 5000 |
| master-read-only | Sets the instance to be read-only. All write operations will fail. | • yes<br>• no | no |
| maxclients | The maximum number of clients allowed to be concurrently connected to a DCS instance. | 1000–10,000 | 10,000 |
| proto-max-bulk-len | Maximum size of a single element request (in bytes). | 1,048,576–536,870,912 | 536,870,912 |
| repl-backlog-size | The replication backlog size (bytes). The backlog is a buffer that accumulates replica data when replicas are disconnected from the master. When a replica reconnects, a partial synchronization is performed to synchronize the data that was missed while replicas were disconnected. | 16,384–1,073,741,824 | 1,048,576 |
| repl-backlog-ttl | The amount of time, in seconds, before the backlog buffer is released, starting from the last a replica was disconnected. The value **0** indicates that the backlog is never released. | 0–604,800 | 3600 |
| repl-timeout | Replication timeout (in seconds). | 30–3600 | 60 |
| hash-max-ziplist-entries | The maximum number of hashes that can be encoded using ziplist, a data structure optimized to reduce memory use. | 1–10,000 | 512 |

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| 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 | If a set is composed entirely of strings that are integers in radix 10 within the range of 64 bit signed integers, the set is encoded using intset, a data structure optimized for memory use. | 1–10,000 | 512 |
| zset-max-ziplist-entries | The maximum number of sorted sets that can be encoded using ziplist, a data structure optimized to reduce memory use. | 1–10,000 | 128 |
| zset-max-ziplist-value | The largest value allowed for a sorted set encoded using ziplist, a special data structure optimized for memory use. | 1–10,000 | 64 |

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| latency-monitor-threshold | The minimum amount of latency that will be logged as latency spikes<br>• Set to 0: Latency monitoring is disabled.<br>• Set to more than 0: All with at least this many ms of latency will be logged.<br>By running the **LATENCY** command, you can perform operations related to latency monitoring, such as obtaining statistical data, and configuring and enabling latency monitoring. For more information about the latency-monitor-threshold, visit **https://redis.io/topics/latency-monitor**. | 0–86,400,000 ms | 0 |

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| notify-keyspace-events | Controls which keyspace events notifications are enabled for. If this parameter is configured, the Redis Pub/Sub feature will allow clients to receive an event notification when a Redis data set is modified. | A string of different values can be used to enable notifications for multiple event types: Possible values include:<br>K: Keyspace events, published with the __keyspace@__ prefix<br>E: Keyevent events, published with __keyevent@__ prefix<br>g: Generic commands (non-type specific) such as DEL, EXPIRE, and RENAME<br>$: String commands<br>l: List commands<br>s: Set commands<br>h: Hash commands<br>z: Sorted set commands<br>x: Expired events (events generated every time a key expires)<br>e: Evicted events (events generated when a key is evicted from maxmemory)<br>For more information, see the following note. | Ex |
| slowlog-log-slower-than | The maximum amount of time allowed, in microseconds, for command execution. If this threshold is exceeded, Redis Slow Log will record the command. | 0–1,000,000 | 10,000 |
| slowlog-max-len | The maximum allowed length of the Redis Slow Log logs. Slow Log consumes memory, but you can reclaim this memory by running the SLOWLOG RESET command. | 0–1,000 | 128 |

NOTE

1. For more information about the parameters described in **Table 7-1**, visit **https://redis.io/topics/memory-optimization**.
2. The **latency-monitor-threshold** parameter is usually used for fault location. After locating faults based on the latency information collected, change the value of **latency-monitor-threshold** to **0** to avoid unnecessary latency.
3. More about the **notify-keyspace-events** parameter:
   - The parameter setting must contain at least a **K** or **E**.
   - **A** is an alias for "g$lshzxe" and cannot be used together with any of the characters in "g$lshzxe".
   - For example, the value **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.

**Table 7-2** DCS Memcached instance configuration parameters

| Parameter | Description | Value Range | Default Value |
| --- | --- | --- | --- |
| timeout | The maximum amount of time (in seconds) a connection between a client and the DCS instance can be allowed to remain idle before the connection is terminated. A setting of **0** means that this function is disabled. | 0–7200 seconds | 0 |
| maxclients | The maximum number of clients allowed to be concurrently connected to a DCS instance. | 1,000–10,000 | 10,000 |
| maxmemory-policy | The policy applied when the maxmemory limit is reached.<br>For more information about this parameter, see **https://redis.io/topics/lru-cache**. | volatile-lru<br>allkeys-lru<br>volatile-random<br>allkeys-random<br>volatile-ttl<br>noeviction | noeviction |
| reserved-memory-percent | Percentage of the maximum available memory reserved for background processes, such as data persistence and replication. | 0–80 | 30 |

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

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

**----End**

# 7.3 Modifying Maintenance Time Window

On the DCS console, after creating a DCS instance, you can modify the maintenance time window of the DCS instance on the instance's **Basic Information** page.

## Prerequisites

At least one DCS instance has been created.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

**Step 4** Click the name of the DCS instance for which you want to modify the maintenance time window.

**Step 5** Click the **Basic Information** tab. In the **Instance Details** area, click the  icon next to the **Maintenance** parameter.

**Step 6** Select a new maintenance time window from the drop-down list. Click  to save the modification or  to discard the modification.

The modification will take effect immediately, that is, the new maintenance time window will appear on the **Basic Information** tab page immediately.

**----End**

# 7.4 Modifying the Security Group

On the DCS console, after creating a DCS instance, you can modify the security group of the DCS instance on the instance's **Basic Information** page.

You can modify the security groups of DCS Redis 3.0 instances but cannot modify those of DCS Redis 4.0 or 5.0, or Redis 6.0 professional edition instances.

## Prerequisites

At least one DCS instance has been created.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

**Step 4** Click the name of the DCS instance for which you want to modify the security group.

**Step 5** Click the **Basic Information** tab. In the **Network** area, click next to the **Security Group** parameter.

**Step 6** Select a new security group from the drop-down list. Click to save the modification or to discard the modification.

NOTE

Only the security groups that have been created can be selected from the drop-down list. If you need to create a security group, follow the procedure described in **How Do I Configure a Security Group?**

The modification will take effect immediately, that is, the new maintenance time window will appear on the **Basic Information** tab page immediately.

**----End**

## 7.5 Viewing Background Tasks

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

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

Filter DCS instances to find the DCS instance whose background tasks you want to manage. You can filter by instance status and name.

**Step 4** Click the name of the DCS instance whose background task you want to manage.

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

A list of background tasks is displayed.

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

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

  NOTE

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

**----End**

# 7.6 Viewing Data Storage Statistics of a Proxy Cluster Instance

You can view the data storage statistics of all nodes of a Proxy cluster instance. If data storage is unevenly distributed across nodes, you can scale up the instance or clear data.

You can only view data storage statistics of Proxy Cluster instances. Instances of other types only have one node, and you can view the used memory on the instance details page.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

On the **Cache Manager** page, you can filter DCS instances by instance status and/or name to find the DCS instance whose data you want to restore.

**Step 4** Click the name of a DCS Redis cluster instance to view the basic information.

**Step 5** Click the **Node Management** tab.

The data volume of each node in the cluster instance is displayed.

When the data storage capacity of a node in a cluster is used up, you can scale up the instance according to **6.2 Modifying Specifications**.

**----End**

# 7.7 Managing Tags

Tags facilitate DCS instance identification and management.

You can add tags to an instance when creating it or add, modify, or delete tags on the details page of a created instance. Each instance can have a maximum of 10 tags.

A tag consists of a tag key and a tag value. **Table 7-3** lists the tag key and value requirements.

**Table 7-3** Tag key and value requirements

| Parameter | Requirements |
| --- | --- |
| Tag key | • Cannot be left blank.<br>• Must be unique for the same instance.<br>• Consists of a maximum of 36 characters.<br>• Cannot contain the following characters: =*<>\,\|/<br>• Cannot start or end with a space. |
| Tag value | • Consists of a maximum of 43 characters.<br>• Cannot contain the following characters: =*<>\,\|/<br>• Cannot start or end with a space. |

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of the instance you want to view.

**Step 5** Click the **Tags** tab.

View the tags of the instance.

**Step 6** Perform the following operations as required:

- Add a tag
  a. Click **Add 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
  In the same row as the tag to be modified, choose **Operation** > **Edit**. Enter the new tag value and click **OK**.

- Delete a tag

  In the same row as the tag to be deleted, choose **Operation** > **Delete**. Then click **Yes**.

**----End**

# 7.8 Managing Shards and Replicas

This section describes how to query the shards and replicas of a DCS Redis 4.0 or 5.0, or Redis 6.0 professional edition instance and how to manually promote a replica to master.

Currently, **this function is supported only by master/standby, Proxy Cluster, and Redis Cluster DCS Redis 4.0 or 5.0 instances, and master/standby DCS Redis 6.0 professional edition instances. DCS Redis 3.0 instances do not support this function.**

A master/standby instance has only one shard with one master and one replica. You can only view the sharding information. To manually switch the master and replica roles, see **6.6 Performing a Master/Standby Switchover**.

A Proxy Cluster or Redis Cluster instance has multiple shards. Each shard has one master and one replica. You can view the sharding information and manually switch the master and replica roles. For details about the number of shards for different instance specifications, see **Proxy Cluster DCS Redis 4.0 and 5.0 Instances** and **Redis Cluster DCS Redis 4.0 and 5.0 Instances**.

## Promoting a Replica to Master

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

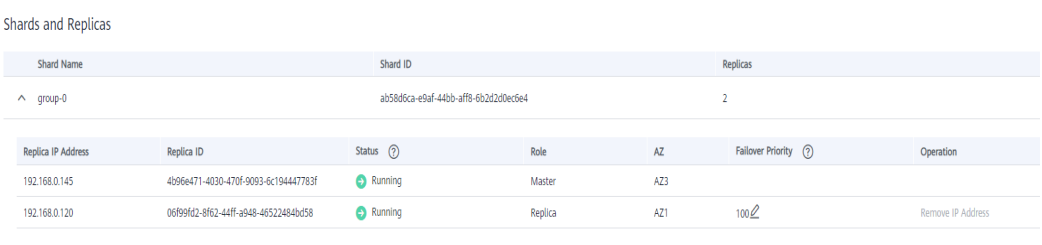
**Step 4** Click an instance.

**Step 5** Click the **Shards and Replicas** tab.

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

**Step 6** Click to show all replicas of a shard.

**Figure 7-1** Lists of shards and replicas



**Step 7** Click **Promote to Master** in the row containing another replica whose role is **Replica**.

**Step 8** Click **Yes**.

**----End**

# 7.9 Cache Analysis

## 7.9.1 Analyzing Big Keys and Hot Keys

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

**Notes on big key analysis:**

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

**Notes on hot key analysis:**

- Only DCS Redis 4.0 and 5.0 instances and Redis 6.0 professional edition instances support hot key analysis, and the **maxmemory-policy** parameter of the instances must be set to **allkeys-lfu** or **volatile-lfu**.
- During hot key analysis, all keys will be traversed. The larger the number of keys, the longer the analysis takes.
- Perform hot key analysis shortly after peak hours to ensure the accuracy of the analysis results.
- The hot key analysis is performed on the master node of each instance and will reduce the instance access performance by up to 10%.
- A maximum of 100 analysis records are retained for each instance. When this limit is reached, the oldest records will be deleted to make room for new records. You can also manually delete records you no longer need.

**NOTE**

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

## Big Key Analysis Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of a DCS Redis instance.

**Step 5** Click the **Cache Analysis** tab.

**Step 6** On the **Big Key Analysis** tab page, manually perform big key analysis or schedule daily automatic analysis.

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

You can view the analysis results of different data types.

NOTE

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

**Figure 7-2** Viewing the results of big key analysis (for Strings)

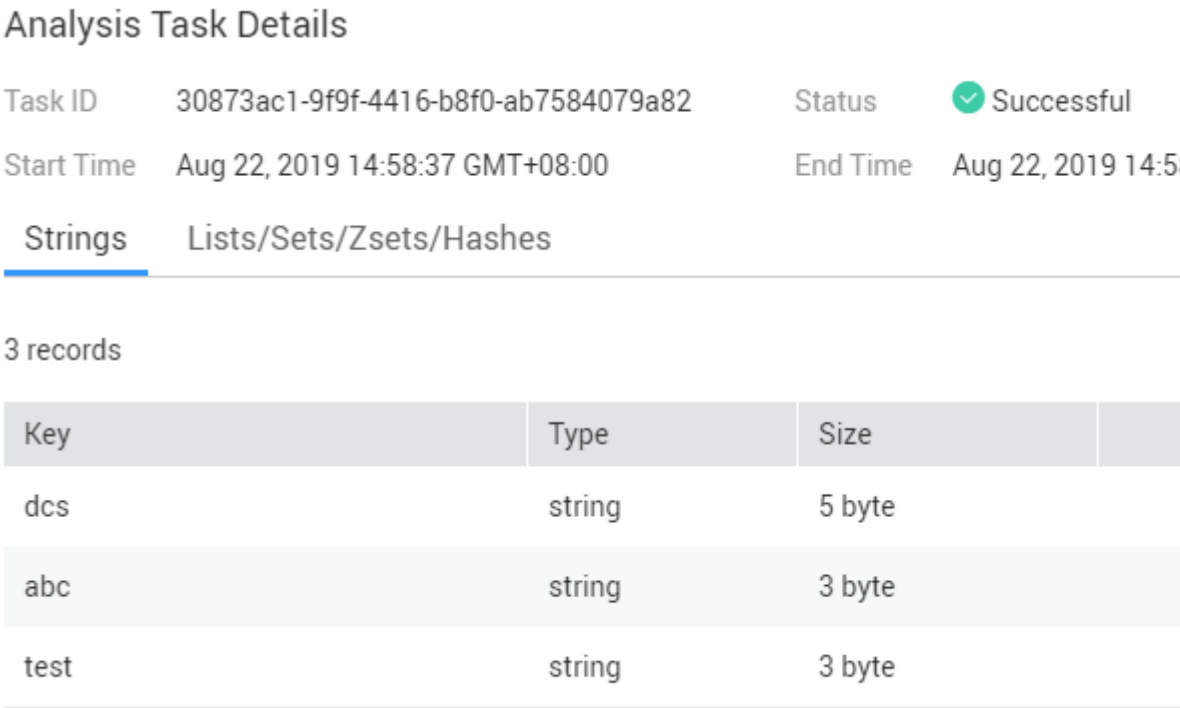
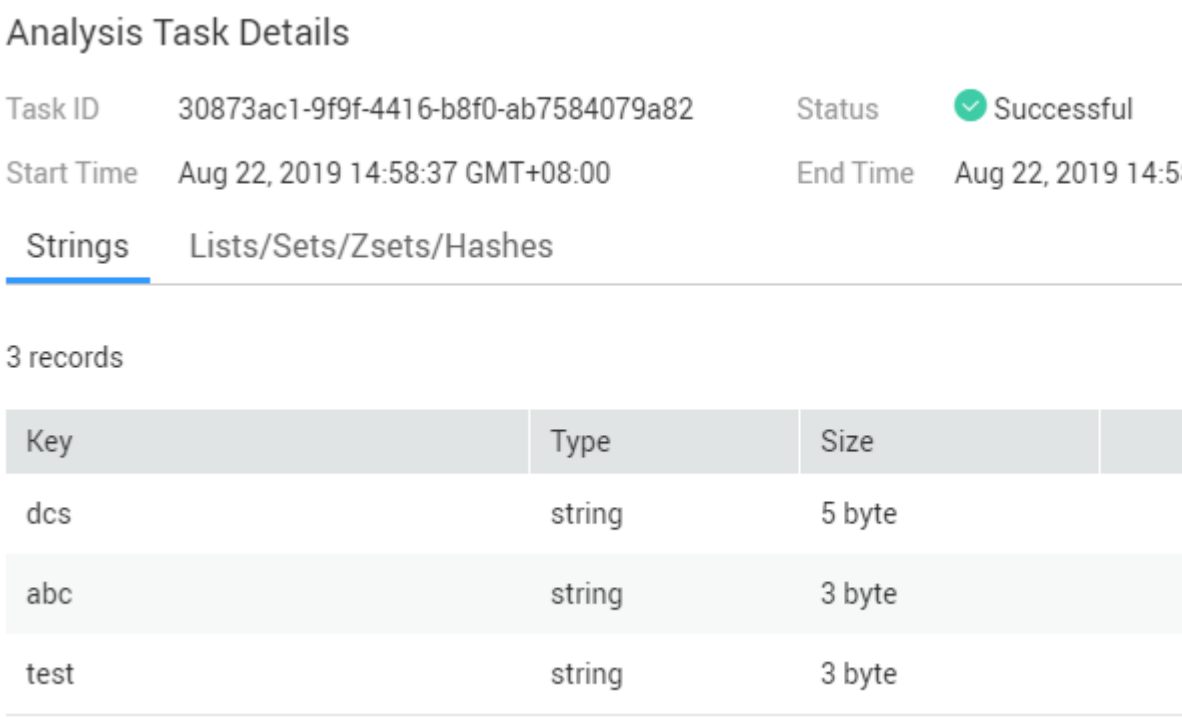


**Figure 7-3** Viewing the results of big key analysis (for Lists/Sets/Zsets/Hashes)




**----End**

## Hot Key Analysis Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of a DCS instance.

**Step 5** Click the **Cache Analysis** tab.

**Step 6** On the **Hot Key Analysis** tab page, manually perform hot key analysis or schedule daily automatic analysis.

NOTE

If the instance was created before July 2020, the default value of the **maxmemory-policy** parameter is **noeviction**. Before starting hot key analysis, set this parameter to **allkeys-lfu** or **volatile-lfu**. If the instance was created after July 2020, the default value of the **maxmemory-policy** parameter is **volatile-lru**. You can start hot key analysis immediately.

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

The hot key analysis results are displayed.

NOTE

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

**Figure 7-4** Viewing the results of hot key analysis

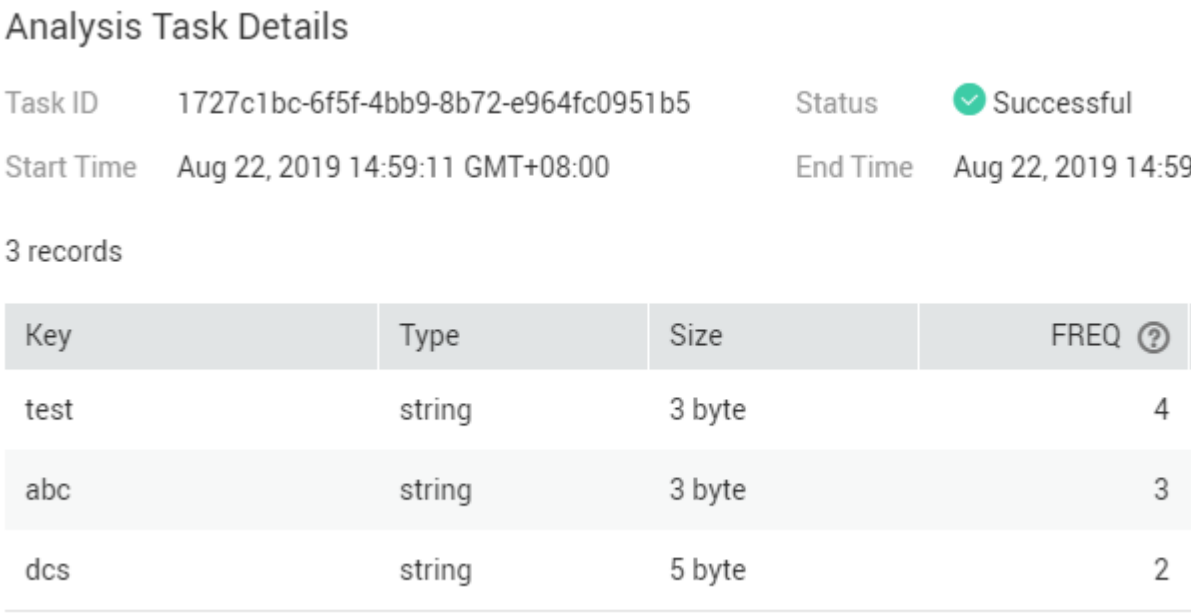



**Table 7-4** Results of hot key analysis

| Parameter | Description |
| --- | --- |
| Key | Name of a hot key. |
| Type | Type of a hot key, which can be string, hash, list, set, or sorted set. |
| Size | Size of the hot key value. |
| FREQ | Reflects the access frequency of a key within a specific period of time.<br>**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. |
| DataBase | Database where a hot key is located. |

**----End**

### FAQs About Big Key and Hot Key

- **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 Big Key?**
- **What Is the Impact of a Hot Key?**
- **How Do I Avoid Big Keys and Hot Keys?**
- **How Do I Analyze the Hot Keys of a DCS Redis 3.0 Instance?**

## 7.9.2 Performing Full Scans

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.

    NOTE

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

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

NOTE

- This function is supported only by DCS Redis 4.0 and 5.0 instances.
- By default, 1000 keys are scanned each time.
- Full scans are performed on the master node and affect the instance performance. Do not perform full scans during high-demand hours.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

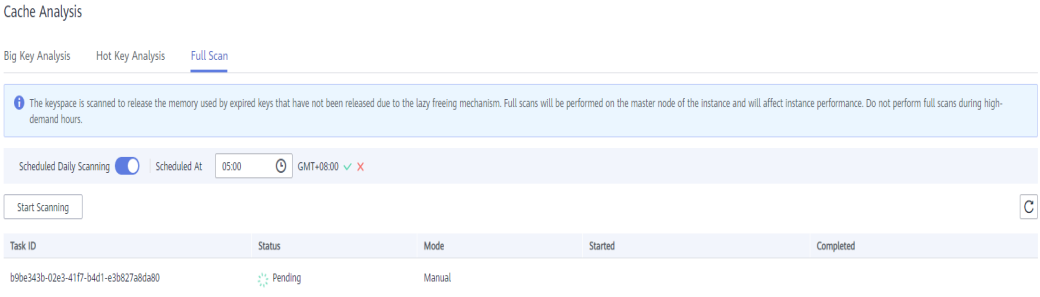
**Step 4** Click a DCS Redis instance.

**Step 5** Click the **Cache Analysis** tab.

**Step 6** On the **Full Scan** tab page, click **Start Scanning** to manually perform a full scan, or schedule daily automatic scanning.

**Step 7** After the full scan task is submitted, you can view it in the task list.

**Figure 7-5** Full scan task lit




**----End**

# 7.10 Managing IP Address Whitelist

DCS helps you control access to your DCS instances in the following modes:

- To control access to Redis 3.0 and Memcached instances, you can use security groups. Whitelists are not supported. For details on how to configure a security group, see **How Do I Configure a Security Group?**
- To control access to Redis 4.0 and 5.0 instances or Redis 6.0 professional edition instances, you can use whitelists. Security groups are not supported.

The following describes how to manage whitelists of a Redis 4.0 or 5.0 instance or a Redis 6.0 professional edition instance to allow access only from whitelisted IP addresses. If no whitelists are added for the instance or the whitelist function is disabled, all IP addresses that can communicate with the VPC can access the instance.

## Creating a Whitelist Group

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of a DCS instance.

**Step 5** Click the **Whitelist** tab and then click **Create Whitelist Group**.

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

**Table 7-5** Whitelist parameters

| Parameter | Description | Example |
| --- | --- | --- |
| Group Name | Whitelist group name of the instance.<br>A maximum of four whitelist groups can be created for each instance. | DCS-test |
| IP Address/ Range | A maximum of 20 IP addresses or IP address ranges can be added to an instance. Separate multiple IP addresses or IP address ranges with commas.<br>Unsupported IP address and IP address range: 0.0.0.0 and 0.0.0/0. | 10.10.10.1,10.10.10.10 |

**Step 7** Click **OK**.

A whitelist group is automatically enabled for the instance once created. Only whitelisted IP addresses can access the instance.

NOTE

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

**----End**

# 7.11 Viewing Redis Slow Logs

Redis uses slow logs to record 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 the slow log with the following parameters:

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

For details about the configuration parameters, see **7.2.1 Modifying Configuration Parameters of an Instance**.

NOTE

You can view the slow log of a Proxy Cluster DCS Redis 3.0 instance only if the instance is created after October 14, 2019. If the instance was created earlier, submit a service ticket to upgrade it. The upgrade adds the slow log function to the console, and does not affect services.

## Viewing Slow Logs on the Console

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

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

**Step 4** Click the name of a DCS instance.

**Step 5** Click the **Slow Log** tab.

**Step 6** Select a start date and an end date to view slow log records within the specified period.

NOTE

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

**Figure 7-6** Slow log records of an instance


**----End**

# 7.12 Viewing Redis Run Logs

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

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner and select a region and a project.

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

**Step 4** Click a DCS instance.

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

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

If the instance is the master/standby 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.

**----End**

# 7.13 Diagnosing an Instance

## Scenario

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

## Restrictions

- DCS Redis 3.0 and Memcached instances do not support diagnosis.
- Only single-node, master/standby, and Redis Cluster DCS Redis 4.0 and 5.0 instances support diagnosis, while the Proxy Cluster type does not.
- Master/standby DCS Redis 6.0 professional edition instances also support diagnosis.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner and select a region and a project.

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

**Step 4** Click a DCS instance.

**Step 5** Click the **Instance Diagnosis** tab.

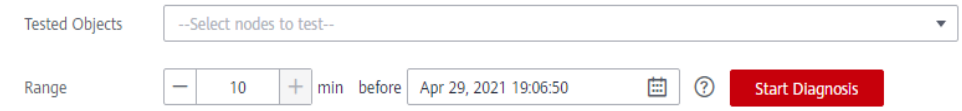
  

**Step 6** Specify the tested object and time range, and click **Start Diagnosis**.

- **Tested Object**: You can select a single node or all nodes. By default, all nodes are tested.
- **Range**: You can specify up to 10 minutes before a point in time in the last 7 days.

  In the following figure, the instance data between 18:03:37 and 18:13:37 on January 7, 2021 will be diagnosed.

  **Figure 7-7** Specifying the tested object and time range




**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**

# 8 Backing Up and Restoring Instances

## 8.1 Overview

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

### Importance of DCS Instance Backup

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

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

### Backup Modes

DCS instances support the following backup modes:

- Scheduled 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 scheduled 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 scheduled 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 also be issued manually. Then, data in the chosen DCS instances will be permanently backed up to OBS. Backup data can be deleted manually.

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

## Additional Information About Data Backup

- Instance type
  - Only master/standby and cluster DCS instances can be backed up and restored, while single-node instances cannot. However, you can export data of a single-node instance to an RDB file using redis-cli. For details, see the **data export and backup FAQ**.
  - DCS Memcached instances

    Only master/standby instances can be backed up and restored. Single-node instances do not support backup or restoration.
- Backup mechanisms

  DCS Redis 3.0 instances use AOF files for data persistence, and DCS Redis 4.0 and 5.0 instances use RDB files for data persistence.

  To export RDB backup files of DCS Redis 3.0 instances, run the **redis-cli -h {redis_address} -p 6379 [-a password] --rdb {output.rdb}** command in redis-cli.

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

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

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

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

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

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

  Backup files are stored to OBS.

  DCS provides the backup service free of charge, but OBS charges will be incurred for the amount and period that storage space is consumed.
- Handling exceptions in scheduled backup

  If a scheduled backup task is triggered while the DCS instance is restarting or being scaled up, the scheduled 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

  Scheduled backup files are retained for up to seven days. You can configure the retention period. At the end of the retention period, most backup files of

the DCS instance will be automatically deleted, but at least one backup file will be retained.

Manual backup files are retained permanently and need to be manually deleted.

### Data Restoration

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

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

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

## 8.2 Configuring a Backup Policy

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

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

### Prerequisites

At least one master/standby DCS instance has been created.

### Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

On the **Cache Manager** page, you can filter DCS instances by instance status and/or name to find the DCS instance whose data you want to restore.

**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**.

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

**Table 8-1** 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.<br>You can select one or multiple days of a week. |
| Retention Period (days) | The number of days that automatically backed up data is retained.<br>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<br>The 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.<br>**NOTE**<br>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.<br>Only instances in the **Running** state can be backed up. |

**Step 7** Click **OK**.

**----End**

# 8.3 Manually Backing Up a DCS Instance

You need to 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.

By default, manually backed up data is permanently retained. If backup data is no longer in use, you can delete it manually.

## Prerequisites

At least one master/standby DCS instance is in the **Running** state.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

On the **Cache Manager** page, you can filter DCS instances by instance status and/or name 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**.

**Step 6** Click **Create Backup**.

**Step 7** Select a backup file format.

Only DCS Redis 4.0 and 5.0 instances and Redis 6.0 professional edition instances support backup file format selection.

**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**

# 8.4 Restoring a DCS Instance

On the DCS console, you can restore backup data to a chosen DCS instance.

## Prerequisites

- At least one master/standby or cluster 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 **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

Filter DCS instances to find the DCS instance whose backup data you want to restore. You can filter by instance status and name.

**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 **Restore** in the same row as 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**

# 8.5 Downloading an RDB or AOF Backup File

Automatically backed up data can be retained for a maximum of 7 days. Manually backed up data is not free of charge and takes space in OBS. Due to these limitations, you are advised to download the RDB and AOF backup files and permanently save them on the local host.

This function is supported only by master/standby 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?**

To export the data of a master/standby or cluster instance, do as follows:

- Redis 3.0: Export the instance data to AOF files by using the DCS console, or to RDB files by running the **redis-cli -h *{redis_address}* -p 6379 [-a *password*] --rdb {output.rdb}** command by using redis-cli.
- Redis 4.0 and 5.0: Export the instance data to AOF or RDB files by using the DCS console.
- Redis 6.0 professional edition: Export the instance data to AOF or RDB files by using the DCS console.

## Prerequisites

The instance has been backed up and the backup is still valid.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

On the **Cache Manager** page, filter DCS instances by instance status and/or name to find the DCS instance you want to restore.

**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** Select the historical backup data to be downloaded, and click **Download**.

**Step 7** In the displayed, **Download Backup File** dialog box, select either of the following two download methods.

Download methods:

- By URL
  a. Step 1: Set the URL validity period and click **Query**.
  b. Step 2: Download the backup file by using the URL list.

  NOTE

  If you choose to copy URLs, use quotation marks to quote the URLs when running the **wget** command in Linux. For example:

  **wget 'https://obsEndpoint.com:443/redisdemo.rdb?parm01=value01&parm02=value02'**

  This is because the URL contains the special character and (&), which will confuse the **wget** command. Quoting the URL facilitates URL identification.
- By OBS

  Follow the displayed procedure.

**----End**

# 9 Migrating Instance Data

## 9.1 Data Migration Overview

The DCS console supports online migration (in full or incrementally) and backup migration (by importing backup files) with intuitive operations.

- Backup migration is suitable when the source and target Redis instances are not connected, and the source Redis instance does not support the **SYNC** and **PSYNC** commands. To migrate data, import your backup files to OBS, and DCS will read data from OBS and migrate the data to the target DCS Redis instance.
- 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.

For more information about migration tools and schemes, see **Migration Tools and Schemes**.

**Table 9-1** DCS data migration modes

| Migration Mode | Source | Target: DCS | | |
|---|---|---|---|---|
| | | Single-Node and Master/Standby | Proxy Cluster | Redis Cluster |
| **Importing backup files** | OBS bucket: AOF files | √ | √ | × |
| | OBS bucket: RDB files | √ | √ | √ |
| **Migrating data online** | DCS for Redis: single-node or master/standby | √ | √ | √ |
| | DCS for Redis: Proxy Cluster | √ | √ | √ |

| | | | | |
|---|---|---|---|---|
| | DCS for Redis: Redis Cluster | √ | √ | √ |
| | Self-hosted single-node or master/standby Redis | √ | √ | √ |
| | Self-hosted proxy-based cluster Redis | √ | √ | √ |
| | Self-hosted Redis Cluster | √ | √ | √ |
| | Other Redis: single-node or master/standby | × | × | × |
| | Other Redis: proxy-based cluster | × | × | × |
| | Other Redis: Redis Cluster | × | × | × |

NOTE

- **DCS for Redis** refers to Redis instances provided by DCS
- **Self-hosted Redis** refers to self-hosted Redis on the cloud, from other cloud vendors, or in on-premises data centers.
- **Other cloud Redis** refers to Redis services provided by other cloud vendors.
- √: Supported. ×: Not supported.

# 9.2 Importing Backup Files from an OBS Bucket

## Application Scenarios

Use the DCS console to migrate Redis data from Redis of another cloud or self-hosted Redis to HUAWEI CLOUD DCS for Redis.

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

.aof, .rbb, .zip, and .tar.gz files can be uploaded to OBS buckets. You can directly upload .aof and .rdb files or compress them into .zip or .tar.gz files before uploading.

## Prerequisites

- The OBS bucket must be in the same region as the target DCS Redis instance. For example, they both reside in the CN North-Beijing1 region.
- The data files to be uploaded must be in the .aof, .rdb, .zip, or .tar.gz format.
- To migrate data from a single-node or master/standby Redis instance of another cloud, create a backup task and download the backup file.
- To migrate data from a cluster Redis instance of another cloud, download all backup files, upload all of them to the OBS bucket, and select all of them for the migration. Each backup file contains data for a shard of the instance.
- .rdb backup files of self-hosted Redis 5.0 cannot be imported. .rdb backup files of self-hosted Redis 3.0 or 4.0 can be exported using redis-cli. .rdb files of other cloud Redis can be exported only by creating backup tasks, and cannot be exported by running commands in redis-cli.
- Redis Cluster instances only support .rdb files.

## Step 1: Prepare the Target DCS Redis Instance

- If a target DCS Redis instance is not available, create one first. For details, see **Buying a DCS Redis Instance**.
- If you already have a DCS Redis instance, you do not need to create one again, but you need to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**.

You can use a DCS Redis 3.0, 4.0, or 5.0 instance as the target instance.

## Step 2: Create an OBS Bucket and Upload Backup Files

**Step 1** Upload the backup data files to the OBS bucket by using OBS Browser+.

If the backup file to be uploaded is smaller than 5 GB, go to step **Step 2** to upload the file using the OBS console.

If the backup file to be uploaded is larger than 5 GB, follow the **instructions** provided by OBS.

**Step 2** On the OBS console, upload the backup data files to the OBS bucket.

Perform the following steps if the backup files are smaller than 5 GB:

1. Create an OBS bucket.

   When creating an OBS bucket, pay attention to the configuration of the following parameters. For details on how to set other parameters, see **Creating a Bucket** in *OBS User Guide*.

   a. **Region**:

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

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

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

   c. Click **Create Now**.

2. In the bucket list, click the bucket created in **Step 2.1**.
3. In the navigation pane, choose **Objects**.
4. On the **Objects** tab page, click **Upload Object**.
5. Specify **Storage Class**.

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

   Drag files or folders to the **Upload Object** area or click **add file**.

   A maximum of 100 files can be uploaded at a time. The total size cannot exceed 5 GB.

   **Figure 9-1** Uploading objects in batches


7. (Optional) Select **KMS encryption** to encrypt the uploaded files.
8. Click **Upload**.

**----End**

## Step 3: Create a Migration Task

**Step 1** Log in to the DCS console.

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

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

**Step 4** Enter the task name and description.

**Step 5** In the **Source Redis** area, select **OBS Bucket** for **Data Source** and then select the OBS bucket to which you have uploaded backup files.

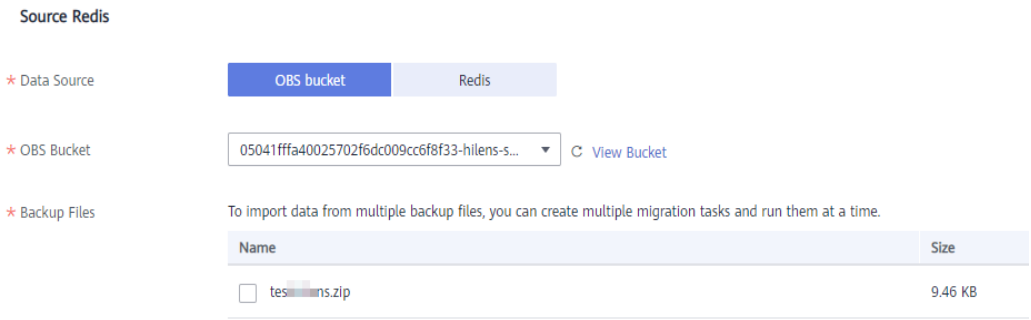
In the **Backup Files** table, the files you have uploaded are displayed.

NOTE

You can upload files in the .aof, .rdb, .zip, or .tar.gz format.

**Figure 9-2** Specifying the backup file information




**Step 6** Select the backup files whose data is to be migrated.

**Step 7** Select the target Redis instance prepared in **Step 1: Prepare the Target DCS Redis Instance**. If the target Redis instance has a password, enter the password and test the connection to check whether the password is correct.

**Step 8** Click **Next**.

**Step 9** 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**

# 9.3 Importing Backup Files from Redis

## Scenario

Use the DCS console to migrate Redis data from self-hosted Redis to DCS for Redis.

Simply back up your Redis data, create a migration task on the DCS console, and then import the backup to a DCS Redis instance.

## Prerequisites

A master/standby or cluster DCS Redis instance has been created as the target for the migration. The source instance has data and has been backed up.

## Step 1: Obtain the Source Instance Name and Password

Obtain the name and password of the source Redis instance.

## Step 2: Prepare the Target DCS Redis Instance

- If a DCS Redis instance is not available, create one first. For details, see **3.3 Buying a DCS Redis Instance**.
- If a DCS Redis instance is available, you do not need to create a new one. However, you must clear the instance data before the migration. For details, see **6.7 Clearing DCS Instance Data**.

You can use a DCS Redis 3.0, 4.0, or 5.0 instance as the target instance.

### Step 3: Create a Migration Task

**Step 1** Log in to the DCS console.

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

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

**Step 4** Enter the task name and description.

**Step 5** Set **Data Source** to **Redis**.

**Step 6** For **Source Redis Instance**, select the instance prepared in **Step 1: Obtain the Source Instance Name and Password**.

**Step 7** Select the backup task whose data is to be migrated.

**Step 8** Select the target instance created in **Step 2: Prepare the Target DCS Redis Instance**.

**Step 9** Enter the password of the target instance. Click **Test Connection** to verify the password.

**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**

## 9.4 Migrating Data Online

### Application Scenarios

If the source and target instances are interconnected and the **SYNC** and **PSYNC** commands are supported by the source instance, data can be migrated online in full or incrementally from the source to the target.

**CAUTION**

- If the **SYNC** and **PSYNC** commands are disabled on the source Redis instance, enable them before performing online migration. Otherwise, the migration fails. If you use a HUAWEI CLOUD DCS Redis instance for online migration, the **SYNC** command is automatically enabled.
- You cannot use public networks for online migration.

### Impacts on Services

During online migration, data is essentially synchronized in full to a new replica. Therefore, perform online migration during low-demand hours.

## Prerequisites

- Before migrating data, read through **Migration Tools and Schemes** to learn about the DCS data migration function and select an appropriate target instance.
- To migrate data from a single-node or master/standby instance to a Redis Cluster instance, check if any data exists in DBs other than DB0 in the source instance. If yes, move the data to DB0. Otherwise, the migration will fail because a Redis Cluster instance has only one DB.

## Step 1: Obtain the Source Redis Address

Obtain the IP address/domain name and port number of the source Redis instance.

## Step 2: Prepare the Target DCS Redis Instance

- If a target DCS Redis instance is not available, create one first. For details, see **Buying a DCS Redis Instance**.
- If you already have a DCS Redis instance, you do not need to create one again, but you need to clear the instance data before the migration. For details, see **Clearing DCS Instance Data**.

  If the target instance data is not cleared before the migration and the source and target instances contain the same key, the key in the target instance will be overwritten by the key in the source instance after the migration.

## Step 3: Check the Network

**Step 1** Check whether the source Redis instance, the target Redis instance, and the migration task are configured with the same VPC.

If yes, go to **Step 5: Configure 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 **Step 1**. If no, go to **Step 3**.

**Step 3** Perform the following operations to establish the network.

- If the source and target Redis instances are in the same region, create a VPC peering connection by referring to **VPC Peering Connection** .
- If the source and target Redis instances are in different regions, create a cloud connection by referring to **Cloud Connect Getting Started**.
- If the source and target Redis instances are on different clouds, create a connection by referring to **Direct Connect documentation**.

**----End**

## Step 4: Create an Online Migration Task

**Step 1** Log in to the DCS console.

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

**Step 3** Click **Create Online Migration Task**.

**Step 4** Enter the task name and description.

**Step 5** Configure the VPC, subnet, and security group for the migration task.

The VPC, subnet, and security group facilitate the migration. Ensure that the migration resources can access the source and target Redis instances.

**----End**

## Step 5: Configure 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.

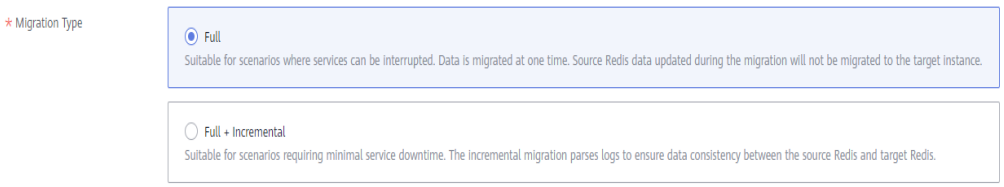
**Step 2** Select a migration type.

Supported migration types are **Full** and **Full + Incremental**, which are described in **Table 9-2**.

**Table 9-2** Migration type description

| Migration Type | Description |
| --- | --- |
| Full | Suitable for scenarios where services can be interrupted. Data is migrated at one time. **Source instance data updated during the migration will not be migrated to the target instance.** |
| Full + incremental | Suitable for scenarios requiring minimal service downtime. The incremental migration parses logs to ensure data consistency between the source and target instances.<br>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. |

**Figure 9-3** Selecting the migration type




**Step 3** Configure source Redis and target Redis.

1. **Source Redis Type**: Select **Redis in the cloud** or **Self-hosted Redis** as required.

- **Redis in the cloud**: a HUAWEI CLOUD DCS Redis instance that is in the same VPC as the migration task
- **Self-hosted Redis**: self-hosted Redis on HUAWEI CLOUD, in another cloud, or in on-premises data centers. If you select this option, enter Redis addresses.

2. If the instance is password-protected, you can click **Test Connection** to check whether the instance password is correct and whether the network is connected.

**Step 4** For **Target Redis Instance**, select the DCS Redis Instance prepared in **Step 2: Prepare the Target DCS Redis Instance**.

If the instance is password-protected, you can click **Test Connection** to check whether the instance password meets the requirements.

**Step 5** Confirm the migration task details and click **Submit**.

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

NOTE

Once incremental migration starts, it remains **Migrating** until you click **Stop**.

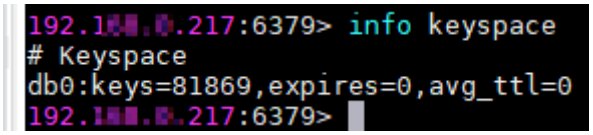
If the migration fails, click the migration task and check the log on the **Migration Logs** page.

**----End**

## Verifying the Migration

After the migration is complete, use redis-cli to connect the source and target Redis instances to check data integrity.

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




3. Calculate the difference between the values of **keys** and **expires** of the source Redis and the target Redis.

   If the differences are the same, the data is complete and the migration is successful.

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

# 10 Managing Passwords

## 10.1 DCS Instance Passwords

Passwords can be configured to control access to your DCS instances, ensuring the security of your data.

You can set a password during or after instance creation. For details on how to set a password after an instance has been created, see **Resetting Instance Passwords**.

You can choose whether to enable password-free access based on your security and convenience trade-off.

### Scenarios Requiring Passwords

- For a DCS instance that is used on the live network or contains important information, you are advised to set a password.
- For a DCS instance with public access enabled, a password must be set to ensure data security.

For details on how to access an instance with a password, see **Accessing a DCS Instance**.

### Using Passwords Securely

1. Hide the password when using redis-cli.

   If the **-a <password>** option is used in redis-cli in Linux, the password is prone to leakage because it is logged and kept in the history. You are advised not to use the **-a <password>** option when running commands in redis-cli. After you have connected to Redis, run the **auth** command to complete authentication, as shown in the following example:

   ```
   $ redis-cli -h 192.168.0.148 -p 6379
   redis 192.168.0.148:6379>auth yourPassword
   OK
   redis 192.168.0.148:6379>
   ```

2. Use interactive password authentication or switch between users with different permissions.

If the script involves DCS instance access, use interactive password authentication. To enable automatic script execution, manage the script as another user and authorize execution using sudo.

3. Use an encryption module in your application to encrypt the password.

# 10.2 Changing Instance Passwords

On the DCS console, you can change the password required for accessing your DCS instance.

NOTE

- You cannot change the password of a DCS instance in password-free mode.
- The DCS instance for which you want to change the password is in the **Running** state.
- The new password will not take effect in the client until the client is restarted.

## Prerequisites

At least one DCS instance has been created.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

**Step 4** Choose **More** > **Change Password** in the same row as the chosen instance.

**Step 5** In the displayed dialog box, set **Old Password**, **New Password**, and **Confirm Password**.

NOTE

After 5 consecutive incorrect password attempts, the account for accessing the chosen DCS instance will be locked for 5 minutes. Passwords cannot be changed during the lockout period.

The password must meet the following requirements:

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

**Step 6** In the **Change Password** dialog box, click **OK** to confirm the password change.

**----End**

# 10.3 Resetting Instance Passwords

On the DCS console, you can configure a new password if you forget your instance password.

NOTE

- For a DCS Redis or Memcached instance, you can change it from password mode to password-free mode or from password-free mode to password mode by resetting its password. For details, see **10.5 Changing Password Settings for DCS Memcached Instances**.
- The DCS instance for which you want to reset the password is in the **Running** state.

## Prerequisites

At least one DCS instance has been created.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

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

**Step 4** Choose **More** > **Reset Password** in the same row as the chosen instance.

**Step 5** In the displayed dialog box, set **New Password**, and **Confirm Password**.

NOTE

The password must meet the following requirements:

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

**Step 6** Click **OK**.

NOTE

The system will display a success message only after the password is successfully reset on all nodes. If the reset fails, the instance will restart and the password of the cache instance will be restored.

**----End**

# 10.4 Changing Password Settings for DCS Redis Instances

## Scenario

DCS Redis instances can be accessed with or without passwords. After an instance is created, you can change its password setting in the following scenarios:

- To enable public access for a password-free DCS Redis instance, you must change the instance to password-protected mode before enabling public access.
- To access a DCS Redis instance in password-free mode, you can enable password-free access to clear the existing password of the instance.

NOTE

- To change the password setting, the DCS Redis instance must be in the **Running** state.
- Password-free access may compromise security. You can set a password by using the password reset function.
- For security purposes, password-free access must be disabled when public access is enabled.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

**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 same row as the chosen instance.

**Step 5** In the **Reset Password** dialogue box, perform either of the following operations as required:

- From password-protected to password-free:

  Switch the toggle for **Password-Free Access** and click **OK**.
- From password-free to password-protected:

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

**----End**

# 10.5 Changing Password Settings for DCS Memcached Instances

## Scenario

DCS Memcached instances can be accessed with or without passwords. After an instance is created, you can change its password setting in the following scenarios:

- If you want to access a password-protected DCS Memcached instance without the username and password, you can enable password-free access to clear the username and password of the instance.

  The Memcached text protocol does not support username and password authentication. To access a DCS Memcached instance by using the Memcached text protocol, you must enable password-free access to the instance.
- If you want to access a password-free DCS Memcached instance using a username and password, you can set a password for the instance using the password reset function.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

**Step 4** To enable password-free access to a DCS Memcached instance, choose **Operation** > **More** > **Reset Password** in the same row as the chosen instance.

**Step 5** In the **Reset Password** dialogue box, perform either of the following operations as required:

- From password-protected to password-free:

  Switch the toggle for **Password-Free Access** and click **OK**.
- From password-free to password-protected:

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

**----End**

# 11 Quotas

## What Is Quota?

A quota is a limit on the quantity or capacity of a certain type of service resources that you can use, for example, the maximum number of DCS instances that you can create and the maximum amount of memory that you can use.

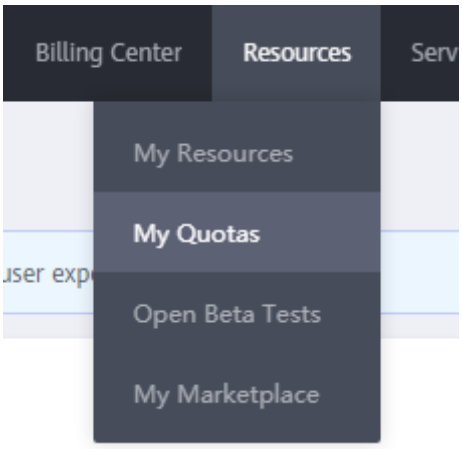
If a quota cannot meet your needs, apply for a higher quota.

## How Do I View My Quota?

1. Log in to the management console.
2. Click in the upper left corner of the management console and select a region and a project.
3. In the upper right corner of the page, choose **Resources** > **My Quotas**.

   The **Service Quota** page is displayed.

   **Figure 11-1** My Quotas

4. On the **Service Quota** page, view the used and total quotas of resources.

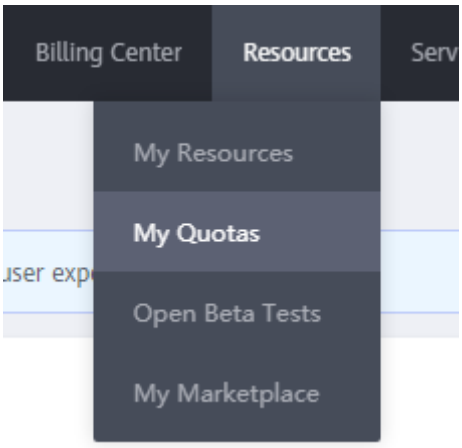
   If a quota cannot meet your needs, apply for a higher quota by performing the following operations.

## How Do I Increase My Quota?

1. Log in to the management console.
2. In the upper right corner of the page, choose **Resources** > **My Quotas**.
   The **Service Quota** page is displayed.

   **Figure 11-2** My Quotas

3. Click **Increase Quota**.
4. On the **Create Service Ticket** page, set the parameters.
   In the **Problem Description** area, enter the required quota and the reason for the quota adjustment.
5. Read the agreements and confirm that you agree to them, and then click **Submit**.

# 12 Monitoring

Cloud Eye is a secure, scalable monitoring platform. It monitors DCS metrics, and sends notifications if alarms are triggered or events occur.

## 12.1 DCS Metrics

### Introduction

This section describes DCS metrics reported to Cloud Eye as well as their namespaces and dimensions. You can use the Cloud Eye console or call **APIs** to query the DCS metrics and alarms.

Different types of instances are monitored on different dimensions.

**Table 12-1** Monitoring dimensions for different instance types

| Instance Type | | Instance Dimension | Redis Server Dimension | Proxy Dimension |
|---|---|---|---|---|
| Single-node | | Yes<br>The monitoring on the instance dimension is conducted on the Redis Server. | N/A | N/A |
| Master/standby | | Yes<br>The monitoring covers the master node. | Yes<br>The monitoring covers the master and standby nodes. | N/A |
| Cluster | Proxy Cluster | Yes<br>The monitoring covers the aggregated master node data. | Yes<br>The monitoring covers each shard. | Yes<br>The monitoring covers each proxy. |

| Instance Type | | Instance Dimension | Redis Server Dimension | Proxy Dimension |
|---|---|---|---|---|
| | Redis Cluster | Yes<br>The monitoring covers the aggregated master node data. | Yes<br>The monitoring covers each shard. | N/A |

## Namespace

SYS.DCS

## DCS Redis 3.0 Instance Metrics

NOTE

The **Monitored Objects and Dimensions** column lists instances and dimensions that support the corresponding metrics.

**Table 12-2** DCS Redis 3.0 instance metrics

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: %<br>For a single-node or master/standby instance, this metric indicates the CPU usage of the master node.<br>For a Proxy Cluster instance, this metric indicates the average value of all proxies. | 0–100% | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| net_in_throughput | Network Input Throughput | Inbound throughput per second on a port<br>Unit: byte/s | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| net_out_throughput | Network Output Throughput | Outbound throughput per second on a port<br>Unit: byte/s | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| connected_clients | Connected Clients | Number of connected clients (excluding those from slave nodes) | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| client_longest_out_list | Client Longest Output List | Longest output list among current client connections | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| client_biggest_in_buf | Client Biggest Input Buf | Maximum input data length among current client connections<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| blocked_clients | Blocked Clients | Number of clients suspended by block operations such as BLPOP, BRPOP, and BRPOPLPUSH | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory | Used Memory | Number of bytes used by the Redis server<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| used_memory_rss | Used Memory RSS | Resident set size (RSS) memory that the Redis server has used, which is the memory that actually resides in the memory, including all stack and heap memory but not swapped-out memory<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_peak | Used Memory Peak | Peak memory consumed by Redis since the Redis server last started<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_lua | Used Memory Lua | Number of bytes used by the Lua engine<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| memory_frag_ratio | Memory Fragmentation Ratio | Current memory fragmentation, which is the ratio between **used_memory_rss**/ **used_memory**. | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| total_connections_received | New Connections | Number of connections received during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| total_commands_processed | Commands Processed | Number of commands processed during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| total_net_input_bytes | Network Input Bytes | Number of bytes received during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| total_net_output_bytes | Network Output Bytes | Number of bytes sent during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: kbit/s | ≥ 0 kbits/s | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: kbit/s | ≥ 0 kbits/s | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| rejected_connections | Rejected Connections | Number of connections that have exceeded maxclients and been rejected during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expired_keys | Expired Keys | Number of keys that have expired and been deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| evicted_keys | Evicted Keys | Number of keys that have been evicted and deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_hits | Keyspace Hits | Number of successful lookups of keys in the main dictionary during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| keyspace_misses | Keyspace Misses | Number of failed lookups of keys in the main dictionary during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| pubsub_channels | PubSub Channels | Number of Pub/Sub channels | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| pubsub_patterns | PubSub Patterns | Number of Pub/Sub patterns | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_hits_perc | Hit Rate | Ratio of the number of Redis cache hits to the number of lookups<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| command_max_delay | Maximum Command Latency | Maximum latency of commands<br>Unit: ms | ≥ 0 ms | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| auth_errors | Authentication Failures | Number of failed authentications | ≥ 0 | Monitored object:<br>Single-node or master/standby DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| is_slow_log_exist | Slow Query Logs | Existence of slow query logs in the instance | • **1**: yes<br>• **0**: no | Monitored object:<br>Single-node or master/standby DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keys | Keys | Number of keys in Redis | ≥ 0 | Monitored object:<br>Single-node or master/standby DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

## DCS Redis 4.0 and 5.0 Instance Metrics

NOTE

The **Monitored Objects and Dimensions** column lists instances and dimensions that support the corresponding metrics.

**Table 12-3** DCS Redis 4.0 and 5.0 instance metrics

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Single-node or master/ standby DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| command_max_delay | Maximum Command Latency | Maximum latency of commands<br>Unit: ms | ≥ 0 ms | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| total_connections_received | New Connections | Number of connections received during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| is_slow_log_exist | Slow Query Logs | Existence of slow query logs in the instance | • **1**: yes<br>• **0**: no | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expires | Keys With an Expiration | Number of keys with an expiration in Redis | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_hits_perc | Hit Rate | Ratio of the number of Redis cache hits to the number of lookups<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| used_memory | Used Memory | Total number of bytes used by the Redis server<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_dataset | Used Memory Dataset | Dataset memory that the Redis server has used<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_dataset_perc | Used Memory Dataset Ratio | Percentage of data memory that Redis has used to the total used memory<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| used_memory_rss | Used Memory RSS | Resident set size (RSS) memory that the Redis server has used, which is the memory that actually resides in the memory, including all stack and heap memory but not swapped-out memory<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_misses | Keyspace Misses | Number of failed lookups of keys in the main dictionary during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| keys | Keys | Number of keys in Redis | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| blocked_clients | Blocked Clients | Number of clients suspended by block operations | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| connected_clients | Connected Clients | Number of connected clients (excluding those from slave nodes) | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| del | DEL | Number of **DEL** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| evicted_keys | Evicted Keys | Number of keys that have been evicted and deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expire | EXPIRE | Number of **EXPIRE** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expired_keys | Expired Keys | Number of keys that have expired and been deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| get | GET | Number of **GET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| hdel | HDEL | Number of **HDEL** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hget | HGET | Number of **HGET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hmget | HMGET | Number of **HMGET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hmset | HMSET | Number of **HMSET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| hset | HSET | Number of **HSET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| memory_frag_ratio | Memory Fragmentation Ratio | Ratio between Used Memory RSS and Used Memory | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| mget | MGET | Number of **MGET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| mset | MSET | Number of **MSET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| pubsub_channels | PubSub Channels | Number of Pub/Sub channels | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| pubsub_patterns | PubSub Patterns | Number of Pub/Sub patterns | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| set | SET | Number of **SET** commands processed per second | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_lua | Used Memory Lua | Number of bytes used by the Lua engine<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_peak | Used Memory Peak | Peak memory consumed by Redis since the Redis server last started<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| sadd | Sadd | Number of **SADD** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| smembers | Smembers | Number of **SMEMBERS** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| rx_controlled | Flow Control Times | Number of flow control times during the monitoring period<br>If the value is greater than 0, the used bandwidth exceeds the upper limit and flow control is triggered.<br>Unit: Count | ≥ 0 | Monitored object:<br>Redis Cluster instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| bandwidth_usage | Bandwidth Usage | Percentage of the used bandwidth to the maximum bandwidth limit | 0–200% | Monitored object:<br>Redis Cluster instance<br>Dimension:<br>dcs_instance_id | 1 minute |

## Cluster DCS Redis Instance Metrics

NOTE

- The following describes the metrics for cluster DCS instances. For Proxy Cluster instances, the monitoring covers Redis Servers and Proxies. For Redis Cluster DCS Redis 4.0 and 5.0 instances, the monitoring only covers Redis Servers.
- The **Monitored Objects and Dimensions** column lists instances and dimensions that support the corresponding metrics.

**Table 12-4** Redis Server metrics

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| connected_clients | Connected Clients | Number of connected clients (excluding those from slave nodes) | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| client_longest_out_list | Client Longest Output List | Longest output list among current client connections | ≥ 0 | Redis Server of cluster DCS Redis 3.0 or 4.0 instance | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| client_biggest_in_buf | Client Biggest Input Buf | Maximum input data length among current client connections<br>Unit: byte | ≥ 0 | Redis Server of cluster DCS Redis 3.0 or 4.0 instance | 1 minute |
| blocked_clients | Blocked Clients | Number of clients suspended by block operations such as BLPOP, BRPOP, and BRPOPLPUSH | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| used_memory | Used Memory | Total number of bytes used by the Redis server<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| used_memory_rss | Used Memory RSS | RSS memory that the Redis server has used, which including all stack and heap memory but not swapped-out memory<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| used_memory_peak | Used Memory Peak | Peak memory consumed by Redis since the Redis server last started<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| used_memory_lua | Used Memory Lua | Number of bytes used by the Lua engine<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| memory_frag_ratio | Memory Fragmentation Ratio | Current memory fragmentation, which is the ratio between **used_memory_rss**/ **used_memory**. | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| total_connections_received | New Connections | Number of connections received during the monitoring period | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| total_commands_processed | Commands Processed | Number of commands processed during the monitoring period | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| total_net_input_bytes | Network Input Bytes | Number of bytes received during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| total_net_output_bytes | Network Output Bytes | Number of bytes sent during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| rejected_connections | Rejected Connections | Number of connections that have exceeded maxclients and been rejected during the monitoring period | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| expired_keys | Expired Keys | Number of keys that have expired and been deleted during the monitoring period | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| evicted_keys | Evicted Keys | Number of keys that have been evicted and deleted during the monitoring period | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| pubsub_channels | PubSub Channels | Number of Pub/Sub channels | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| pubsub_patterns | PubSub Patterns | Number of Pub/Sub patterns | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| keyspace_hits_perc | Hit Rate | Ratio of the number of Redis cache hits to the number of lookups<br>Unit: % | 0–100% | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| command_max_delay | Maximum Command Latency | Maximum latency of commands<br>Unit: ms | ≥ 0 ms | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| is_slow_log_exist | Slow Query Logs | Existence of slow query logs in the node | • **1**: yes<br>• **0**: no | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| keys | Keys | Number of keys in Redis | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 3.0, 4.0, or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| sadd | Sadd | Number of **SADD** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| smembers | Smembers | Number of **SMEMBERS** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| ms_repl_offset | Replication Gap | Data synchronization gap between the master and the replica | - | Monitored object:<br>Replica of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| del | DEL | Number of **DEL** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| expire | EXPIRE | Number of **EXPIRE** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| get | GET | Number of **GET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| hdel | HDEL | Number of **HDEL** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| hget | HGET | Number of **HGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| hmget | HMGET | Number of **HMGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| hmset | HMSET | Number of **HMSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| hset | HSET | Number of **HSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| mget | MGET | Number of **MGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| mset | MSET | Number of **MSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| set | SET | Number of **SET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| rx_controlled | Flow Control Times | Number of flow control times during the monitoring period<br>If the value is greater than 0, the used bandwidth exceeds the upper limit and flow control is triggered.<br>Unit: Count | ≥ 0 | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |
| bandwidth_usage | Bandwidth Usage | Percentage of the used bandwidth to the maximum bandwidth limit | 0–200% | Monitored object:<br>Redis Server of cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_redis_node | 1 minute |

**Table 12-5** Proxy metrics of Proxy Cluster DCS Redis 3.0 instances

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| p_connected_clients | Connected Clients | Number of connected clients | ≥ 0 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| max_rxpck_per_sec | Max. NIC Data Packet Receive Rate | Maximum number of data packets received by the proxy NIC per second during the monitoring period<br>Unit: packages/second | 0–10,000,000 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| max_txpck_per_sec | Max. NIC Data Packet Transmit Rate | Maximum number of data packets transmitted by the proxy NIC per second during the monitoring period<br>Unit: packages/second | 0–10,000,000 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| max_rxkB_per_sec | Maximum Inbound Bandwidth | Largest volume of data received by the proxy NIC per second<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| max_txkB_per_sec | Maximum Outbound Bandwidth | Largest volume of data transmitted by the proxy NIC per second<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| avg_rxpck_per_sec | Average NIC Data Packet Receive Rate | Average number of data packets received by the proxy NIC per second during the monitoring period<br>Unit: packages/second | 0–10,000,000 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| avg_txpck_per_sec | Average NIC Data Packet Transmit Rate | Average number of data packets transmitted by the proxy NIC per second during the monitoring period<br>Unit: packages/second | 0–10,000,000 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| avg_rxkB_per_sec | Average Inbound Bandwidth | Average volume of data received by the proxy NIC per second<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |
| avg_txkB_per_sec | Average Outbound Bandwidth | Average volume of data transmitted by the proxy NIC per second<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 3.0 instance<br>Dimension:<br>dcs_cluster_proxy_node | 1 minute |

**Table 12-6** Proxy metrics of Proxy Cluster DCS Redis 4.0 or 5.0 instances

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| node_status | Proxy Status | Indication of whether the proxy is normal. | • **0**: Normal<br>• **1**: Abnormal | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| p_connected_clients | Connected Clients | Number of connected clients | ≥ 0 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| total_net_input_bytes | Network Input Bytes | Number of bytes received during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |
| total_net_output_bytes | Network Output Bytes | Number of bytes sent during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance<br>Dimension:<br>dcs_cluster_proxy2_node | 1 minute |

## DCS Redis 6.0 Professional Edition Instance Metrics

NOTE

The **Monitored Objects and Dimensions** column lists instances and dimensions that support the corresponding metrics.

**Table 12-7** Metrics for DCS Redis 6.0 professional edition instances

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Single-node or master/ standby DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| command_max_delay | Maximum Command Latency | Maximum latency of commands<br>Unit: ms | ≥ 0 ms | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| total_connections_received | New Connections | Number of connections received during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| is_slow_log_exist | Slow Query Logs | Existence of slow query logs in the instance | • **1**: yes<br>• **0**: no | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expires | Keys With an Expiration | Number of keys with an expiration in Redis | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_hits_perc | Hit Rate | Ratio of the number of Redis cache hits to the number of lookups<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory | Used Memory | Number of bytes used by the Redis server<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| used_memory_dataset | Used Memory Dataset | Dataset memory that the Redis server has used<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_dataset_perc | Used Memory Dataset Ratio | Percentage of data memory that Redis has used to the total memory<br>Unit: % | 0–100% | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_rss | Used Memory RSS | Resident set size (RSS) memory that the Redis server has used, which is the memory that actually resides in the memory, including all stack and heap memory but not swapped-out memory<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keyspace_misses | Keyspace Misses | Number of failed lookups of keys in the main dictionary during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| keys | Keys | Number of keys in Redis | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| blocked_clients | Blocked Clients | Number of clients suspended by block operations | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| connected_clients | Connected Clients | Number of connected clients (excluding those from slave nodes) | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| del | DEL | Number of **DEL** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| evicted_keys | Evicted Keys | Number of keys that have been evicted and deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| expire | EXPIRE | Number of **EXPIRE** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| expired_keys | Expired Keys | Number of keys that have expired and been deleted during the monitoring period | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| get | GET | Number of **GET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hdel | HDEL | Number of **HDEL** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hget | HGET | Number of **HGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| hmget | HMGET | Number of **HMGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hmset | HMSET | Number of **HMSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| hset | HSET | Number of **HSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| memory_frag_ratio | Memory Fragmentation Ratio | Ratio between Used Memory RSS and Used Memory | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| mget | MGET | Number of **MGET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| mset | Mset | Number of **MSET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| pubsub_channels | PubSub Channels | Number of Pub/Sub channels | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| pubsub_patterns | PubSub Patterns | Number of Pub/Sub patterns | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| set | SET | Number of **SET** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| used_memory_lua | Used Memory Lua | Number of bytes used by the Lua engine<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| used_memory_peak | Used Memory Peak | Peak memory consumed by Redis since the Redis server last started<br>Unit: byte | ≥ 0 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| sadd | SADD | Number of **SADD** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| smembers | SMEMBERS | Number of **SMEMBERS** commands processed per second<br>Unit: Count/s | 0–500,000 | Monitored object:<br>Single-node, master/ standby, or cluster DCS Redis instance<br>Dimension:<br>dcs_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| rx_controlled | Flow Control Times | Number of flow control times during the monitoring period<br>If the value is greater than 0, the used bandwidth exceeds the upper limit and flow control is triggered.<br>Unit: Count | ≥ 0 | Monitored object:<br>Redis Cluster instance<br>Dimension:<br>dcs_instance_id | 1 minute |
| bandwidth_usage | Bandwidth Usage | Percentage of the used bandwidth to the maximum bandwidth limit | 0–200% | Monitored object:<br>Redis Cluster instance<br>Dimension:<br>dcs_instance_id | 1 minute |

## DCS Memcached Instance Metrics

**Table 12-8** DCS Memcached instance metrics

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| cpu_usage | CPU Usage | CPU consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| memory_usage | Memory Usage | Memory consumed by the monitored object<br>Unit: % | 0–100% | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| net_in_throughput | Network Input Throughput | Inbound throughput per second on a port<br>Unit: byte/s | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| net_out_throughput | Network Output Throughput | Outbound throughput per second on a port<br>Unit: byte/s | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_connected_clients | Connected Clients | Number of connected clients (excluding those from slave nodes) | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_used_memory | Used Memory | Number of bytes used by Memcached<br>Unit: byte | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_used_memory_rss | Used Memory RSS | RSS memory used that actually resides in the memory, including all stack and heap memory but not swapped-out memory<br>Unit: byte | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_used_memory_peak | Used Memory Peak | Peak memory consumed since the server last started<br>Unit: byte | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_memory_frag_ratio | Memory Fragmentation Ratio | Ratio between Used Memory RSS and Used Memory | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_connections_received | New Connections | Number of connections received during the monitoring period | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_commands_processed | Commands Processed | Number of commands processed during the monitoring period | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_instantaneous_ops | Ops per Second | Number of commands processed per second | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| mc_net_input_bytes | Network Input Bytes | Number of bytes received during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_net_output_bytes | Network Output Bytes | Number of bytes sent during the monitoring period<br>Unit: byte | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_instantaneous_input_kbps | Input Flow | Instantaneous input traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_instantaneous_output_kbps | Output Flow | Instantaneous output traffic<br>Unit: KB/s | ≥ 0 KB/s | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_rejected_connections | Rejected Connections | Number of connections that have exceeded maxclients and been rejected during the monitoring period | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_expired_keys | Expired Keys | Number of keys that have expired and been deleted during the monitoring period | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_evicted_keys | Evicted Keys | Number of keys that have been evicted and deleted during the monitoring period | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_cmd_get | Number of Retrieval Requests | Number of received data retrieval requests | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_cmd_set | Number of Storage Requests | Number of received data storage requests | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_cmd_flush | Number of Flush Requests | Number of received data clearance requests | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_cmd_touch | Number of Touch Requests | Number of received requests for modifying the validity period of data | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_get_hits | Number of Retrieval Hits | Number of successful data retrieval operations | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| mc_get_misses | Number of Retrieval Misses | Number of failed data retrieval operations due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_delete_hits | Number of Delete Hits | Number of successful data deletion operations | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_delete_misses | Number of Delete Misses | Number of failed data deletion operations due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_incr_hits | Number of Increment Hits | Number of successful increment operations | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| mc_incr_misses | Number of Increment Misses | Number of failed increment operations due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_decr_hits | Number of Decrement Hits | Number of successful decrement operations | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_decr_misses | Number of Decrement Misses | Number of failed decrement operations due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_cas_hits | Number of CAS Hits | Number of successful CAS operations | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_cas_misses | Number of CAS Misses | Number of failed CAS operations due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_cas_badval | Number of CAS Values Not Matched | Number of failed CAS operations due to CAS value mismatch | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_touch_hits | Number of Touch Hits | Number of successful requests for modifying the validity period of data | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_touch_misses | Number of Touch Misses | Number of failed requests for modifying the validity period of data due to key nonexistence | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
|---|---|---|---|---|---|
| mc_auth_cmds | Authentication Requests | Number of authentication requests | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_auth_errors | Authentication Failures | Number of failed authentication requests | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_curr_items | Number of Items Stored | Number of stored data items | ≥ 0 | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_command_max_delay | Maximum Command Latency | Maximum latency of commands<br>Unit: ms | ≥ 0 ms | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

| Metric ID | Metric | Description | Value Range | Monitored Object and Dimension | Monitoring Period (Raw Data) |
| --- | --- | --- | --- | --- | --- |
| mc_is_slow_log_exist | Slow Query Logs | Existence of slow query logs in the instance | • **1**: yes<br>• **0**: no | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |
| mc_keyspace_hits_perc | Hit Rate | Ratio of the number of Memcached cache hits to the number of lookups<br>Unit: % | 0–100% | Monitored object:<br>DCS Memcached instance<br>Dimension:<br>dcs_memcached_instance_id | 1 minute |

### Dimensions

| Key | Value |
| --- | --- |
| dcs_instance_id | DCS Redis instance |
| dcs_cluster_redis_node | Redis Server |
| dcs_cluster_proxy_node | Proxy in a Proxy Cluster DCS Redis 3.0 instance |
| dcs_cluster_proxy2_node | Proxy in a Proxy Cluster DCS Redis 4.0 or 5.0 instance |
| dcs_memcached_instance_id | DCS Memcached instance |

## 12.2 Common Metrics

This section describes common Redis metrics.

**Table 12-9** Common metrics

| Metric | Description |
| --- | --- |
| CPU Usage | This metric indicates the instantaneous CPU usage.<br>• For a single-node or master/standby instance, you can view the CPU usage of the instance.<br>• For a Proxy Cluster instance, you can view the CPU usage of the instance and the proxies.<br>• For a Redis Cluster instance, you can only view the CPU usage of the Redis Servers. |
| Memory Usage | This metric indicates the instantaneous memory usage.<br>• For a single-node or master/standby instance, you can view the memory usage of the instance.<br>• For a Proxy Cluster instance, you can view the memory usage of the instance and the proxies.<br>• For a Redis Cluster instance, you can only view the memory usage of the Redis Servers. |
| Connected Clients | This metric indicates the number of instantaneous connected clients, that is, the number of concurrent connections.<br>This metric does not include the number of connections to the standby nodes of master/standby or cluster instances.<br>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.<br>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.<br>• The monitoring data on the instance level shows the aggregated input traffic of all nodes.<br>• The monitoring data on the node level shows the input traffic of the current node. |
| Output Flow | This metric indicates the instantaneous output traffic.<br>• The monitoring data on the instance level shows the aggregated output traffic of all nodes.<br>• The monitoring data on the node level shows the output traffic of the current node. |

| Metric | Description |
| --- | --- |
| Bandwidth Usage | This metric indicates the percentage of the used bandwidth to the maximum bandwidth limit. |
| Commands Processed | This metric indicates the number of commands processed during the monitoring period. The default monitoring period is 1 minute.<br>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.<br>For details about the maximum allowed bandwidth, see the "Maximum/Assured Bandwidth" column of different instance types listed in **DCS Instance Specifications**. |
| Slow Query Logs | This metric indicates whether slow query logs exist on the instance.<br>For details about why slow query logs exist, see **Viewing Redis Slow Logs**. |

# 12.3 Viewing Metrics

The Cloud Eye service monitors the running performance your DCS instances.

## Procedure

**Step 1** Log in to the **DCS console**.

**Step 2** Click  in the upper left corner of the management console and select a region and a project.

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

**Step 4** Click the desired instance.

**Step 5** Choose **Performance Monitoring**. All monitoring metrics of the instance are displayed.

NOTE

You can also click **View Metric** in the **Operation** column on the **Cache Manager** page. You will be redirected to the Cloud Eye console. The metrics displayed on the Cloud Eye console are the same as those displayed on the **Performance Monitoring** page of the DCS console.

**----End**

# 12.4 Configuring Alarm Rules for Critical Metrics

This section describes the alarm rules of some metrics and how to configure the rules. In actual scenarios, configure alarm rules for metrics by referring to the following alarm policies.

## Alarm Policies for DCS Redis Instances

**Table 12-10** DCS Redis instance metrics to configure alarm rules for

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| CPU Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Consider capacity expansion based on the service analysis.<br>The CPU capacity of a single-node or master/standby instance cannot be expanded. If you need larger capacity, use a cluster instance instead.<br>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<br>Number of consecutive periods: 2<br>Alarm severity: Critical | No | Expand the capacity of the instance. |

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Connected Clients | 0–10,000 | Alarm threshold: 8000<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Optimize the connection pool in the service code to prevent the number of connections from exceeding the maximum limit.<br>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.<br>For single-node and master/standby instances, the maximum number of connections allowed is 10,000. You can adjust the threshold based on service requirements. |
| New Connections<br>(Count/min) | 0–10,000 | Alarm threshold: 10,000<br>Number of consecutive periods: 2<br>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.<br>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. |
| Input Flow | > 0 | Alarm threshold: 80% of the assured bandwidth<br>Number of consecutive periods: 2<br>Alarm severity: Major | Yes | Consider capacity expansion based on the service analysis and bandwidth limit.<br>Configure this alarm only for single-node and master/standby DCS Redis 3.0 instances and set the alarm threshold to 80% of the assured bandwidth of DCS Redis 3.0 instances. |

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Output Flow | > 0 | Alarm threshold: 80% of the assured bandwidth<br>Number of consecutive periods: 2<br>Alarm severity: Major | Yes | Consider capacity expansion based on the service analysis and bandwidth limit.<br>Configure this alarm only for single-node and master/standby DCS Redis 3.0 instances and set the alarm threshold to 80% of the assured bandwidth of DCS Redis 3.0 instances. |

## Alarm Policies for DCS Memcached Instances

**Table 12-11** DCS Memcached instance metrics to configure alarm rules for

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| CPU Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Check the service for traffic surge.<br>The CPU capacity of a single-node or master/standby instance cannot be expanded. Analyze the service and consider splitting the service or combine multiple instances into a cluster on the client end. |
| Memory Usage | 0–100 | Alarm threshold: 65<br>Number of consecutive periods: 2<br>Alarm severity: minor | No | Consider expanding the instance capacity. |

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Connected Clients | 0–10,000 | Alarm threshold: 8000<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Optimize the connection pool in the service code to prevent the number of connections from exceeding the maximum limit. |
| New Connections | 0–10,000 | Alarm threshold: 10,000<br>Number of consecutive periods: 2<br>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. |
| Input Flow | > 0 | Alarm threshold: 80% of the assured bandwidth<br>Number of consecutive periods: 2<br>Alarm severity: Major | Yes | Consider capacity expansion based on the service analysis and bandwidth limit.<br>For details about the bandwidth limits of different instance specifications, see **DCS Instance Specifications**. |
| Output Flow | > 0 | Alarm threshold: 80% of the assured bandwidth<br>Number of consecutive periods: 2<br>Alarm severity: Major | Yes | Consider capacity expansion based on the service analysis and bandwidth limit.<br>For details about the bandwidth limits of different instance specifications, see **DCS Instance Specifications**. |

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Authentication Failures | > 0 | Alarm threshold: 0<br>Number of consecutive periods: 1<br>Alarm severity: Critical | - | Check whether the password is entered correctly. |

## Alarm Policies for Redis Server Nodes of Cluster DCS Redis Instances

**Table 12-12** Redis server metrics to configure alarm policies for

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| CPU Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Check the service for traffic surge.<br>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.<br>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 | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Memory Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Check the service for traffic surge.<br>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<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Check whether the number of connections is within the appropriate range. If yes, adjust the alarm threshold. |
| New Connections | 0–10,000 | Alarm threshold: 10,000<br>Number of consecutive periods: 2<br>Alarm severity: minor | - | Check whether **connect** is used. To ensure performance, use persistent connections ("pconnect" in Redis terminology). |
| Slow Query Logs | 0–1 | Alarm threshold: 0<br>Number of consecutive periods: 1<br>Alarm severity: Major | - | Use the slow log function on the console to analyze slow commands. |

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| Bandwidth Usage | 0–200 | Alarm threshold: 90<br>Number of consecutive periods: 2<br>Alarm severity: Major | Yes | Check whether the bandwidth usage increase comes from read services or write services based on the input and output flow.<br>If the bandwidth usage of a single node is high, check whether big keys exist.<br>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.<br>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. |
| Flow Control Times | > 0 | Alarm threshold: 0<br>Number of consecutive periods: 1<br>Alarm severity: Critical | Yes | Consider capacity expansion based on the specification limits, input flow, and output flow.<br>**NOTE**<br>This metric is supported only by Redis 4.0 and 5.0 and not by Redis 3.0. |

## Alarm Policies for Proxy Nodes of Cluster DCS Redis Instances

**Table 12-13** Proxy metrics to configure alarm policies for

| Metric | Normal Range | Alarm Policy | Approach Upper Limit | Handling Suggestion |
| --- | --- | --- | --- | --- |
| CPU Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Critical | Yes | Consider capacity expansion, which will add Proxies. |
| Memory Usage | 0–100 | Alarm threshold: 70<br>Number of consecutive periods: 2<br>Alarm severity: Critical | Yes | Consider capacity expansion, which will add Proxies. |
| Connected Clients | 0–30,000 | Alarm threshold: 20,000<br>Number of consecutive periods: 2<br>Alarm severity: Major | No | Optimize the connection pool in the service code to prevent the number of connections from exceeding the maximum limit. |

## Configuring an Alarm Rule for a Resource Group

Cloud Eye allows you to add DCS instances, Redis Server nodes, and proxy nodes to resource groups and manage instances and alarm rules by group to simplify O&M. For details, see **Creating a Resource Group**.

**Step 1** Create a resource group.

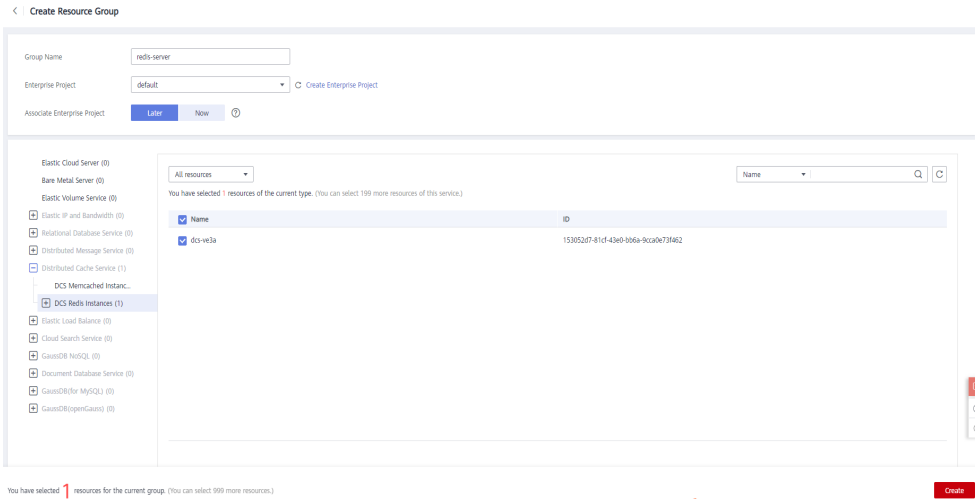
1. Log in to the Cloud Eye console. In the navigation pane, choose **Resource Groups** and then click **Create Resource Group** in the upper right corner.

2. Enter a group name and add Redis Server nodes to the resource group.

   You can add Redis Server nodes of different instances to the same resource group.

   **Figure 12-1** Creating a resource group

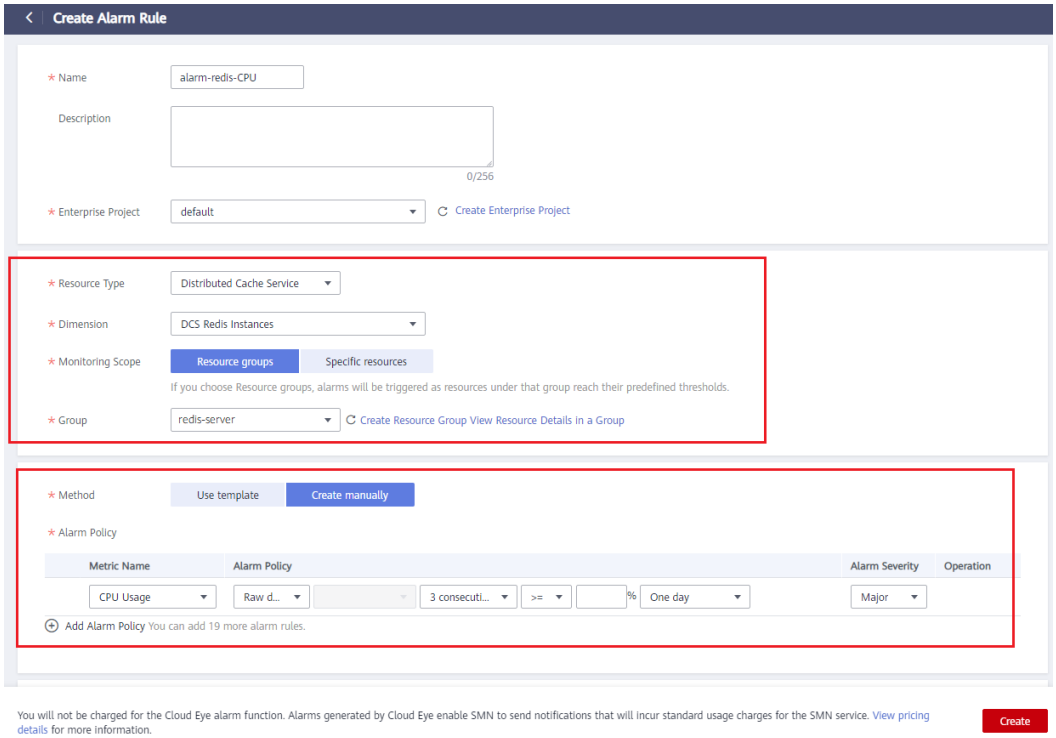



3. Click **Create**.

**Step 2** In the navigation pane of the Cloud Eye console, choose **Alarm Management** > **Alarm Rules** and then click **Create Alarm Rule** to set alarm information for the resource group.

Create a CPU usage alarm rule for all Redis Server nodes in the resource group, as shown in the following figure.

**Figure 12-2** Creating an alarm rule for a resource group



**Step 3** Click **Create**.

**----End**

## Configuring an Alarm Rule for a Specific Resource

In the following example, an alarm rule is set for the **Slow Query Logs** (**is_slow_log_exist**) metric.

**Step 1** Log in to the **DCS console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

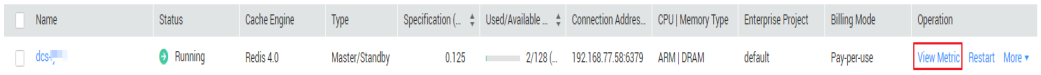
NOTE

Select the same region as your application service.

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

**Step 4** In the same row as the DCS instance whose metrics you want to view, click **View Metric** in the **Operation** column.

**Figure 12-3** Viewing instance metrics




**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. Under **Specify Rules Name**, set the alarm name and description.
2. Under **Alarm Content**, set the alarm policy and severity.

   For example, the alarm policy shown in **Figure 12-4** indicates that an alarm will be triggered if slow logs exist in the instance for two consecutive periods. If no actions are taken, the alarm will be triggered once every day, until the value of this metric returns to **0**.

**Figure 12-4** Setting the alarm content

3. Set **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**.

**----End**

# 13 Auditing

## 13.1 Operations That Can Be Recorded by CTS

With CTS, you can query, audit, and review operations performed on cloud resources. Traces include the operation requests sent using the management console or open APIs as well as the results of these requests.

The following lists the DCS operations that can be recorded by CTS.

**Table 13-1** DCS operations that can be recorded by CTS

| Operation | Resource Type | Trace Name |
| --- | --- | --- |
| Creating an instance | DCS | createDCSInstance |
| Submitting an instance creation request | DCS | submitCreateDCSInstanceRequest |
| Deleting multiple instances | DCS | batchDeleteDCSInstance |
| Deleting an instance | DCS | deleteDCSInstance |
| Modifying instance information | DCS | modifyDCSInstanceInfo |
| Modifying instance configurations | DCS | modifyDCSInstanceConfig |

| Operation | Resource Type | Trace Name |
| --- | --- | --- |
| Changing instance password | DCS | modifyDCSInstancePassword |
| Stopping an instance | DCS | stopDCSInstance |
| Submitting an instance stopping request | DCS | submitStopDCSInstanceRequest |
| Restarting an instance | DCS | restartDCSInstance |
| Submitting an instance restarting request | DCS | submitRestartDCSInstanceRequest |
| Starting an instance | DCS | startDCSInstance |
| Submitting an instance starting request | DCS | submitStartDCSInstanceRequest |
| Clearing instance data | DCS | flushDCSInstance |
| Stopping multiple instances | DCS | batchStopDCSInstance |
| Submitting a request to stop instances in batches | DCS | submitBatchStopDCSInstanceRequest |
| Restarting instances in batches | DCS | batchRestartDCSInstance |
| Submitting a request to restart instances in batches | DCS | submitBatchRestartDCSInstanceRequest |

| Operation | Resource Type | Trace Name |
| --- | --- | --- |
| Starting multiple instances | DCS | batchStartDCSInstance |
| Submitting a request to start instances in batches | DCS | submitBatchStartDCSInstanceRequest |
| Restoring instance data | DCS | restoreDCSInstance |
| Submitting a request to restore instance data | DCS | submitRestoreDCSInstanceRequest |
| Backing up instance data | DCS | backupDCSInstance |
| Submitting a request to back up instance data | DCS | submitBackupDCSInstanceRequest |
| Deleting instance backup files | DCS | deleteInstanceBackupFile |
| Deleting background tasks | DCS | deleteDCSInstanceJobRecord |
| Modifying instance specifications | DCS | modifySpecification |
| Submitting a request to modify instance specifications | DCS | submitModifySpecificationRequest |

| Operation | Resource Type | Trace Name |
| --- | --- | --- |
| Creating an instance subscription order | DCS | createInstanceOrder |
| Creating an order for modifying instance specifications | DCS | createSpecificationChangeOrder |
| Updating enterprise project ID | DCS | updateEnterpriseProjectId |
| Switching between master and standby nodes | DCS | masterStandbySwitchover |
| Disabling public access | DCS | disablePublicNetworkAccess |
| Enabling public access | DCS | enablePublicNetworkAccess |
| Resetting instance password | DCS | resetDCSInstancePassword |
| Submitting a request to clear instance data | DCS | submitFlushDCSInstanceRequest |
| Accessing Web CLI | DCS | webCliLogin |
| Running commands in Web CLI | DCS | webCliCommand |
| Exiting Web CLI | DCS | webCliLogout |
| Migrating offline data | DCS | offlineMigrate |

| Operation | Resource Type | Trace Name |
| --- | --- | --- |
| Changing the billing mode | DCS | billingModeChange |

# 13.2 Viewing Traces on the CTS Console

After CTS is enabled, the tracker starts recording operations on cloud resources. Operation records for the last seven days can be viewed on the CTS console. This section describes how to query operation records of the last seven days on the CTS console.

## Procedure

**Step 1** Log in to the **management console**.

**Step 2** Click in the upper left corner of the management console and select a region and a project.

NOTE

Select the same region as your application service.

**Step 3** Click **Service List** and choose **Management & Deployment** > **Cloud Trace Service**.

**Step 4** In the navigation pane, click **Trace List**.

**Step 5** Specify the filters used for querying traces. The following filters are available:

- **Search By**:

  Select an option from the drop-down list. Select **DCS** from the **Trace Source** drop-down list.

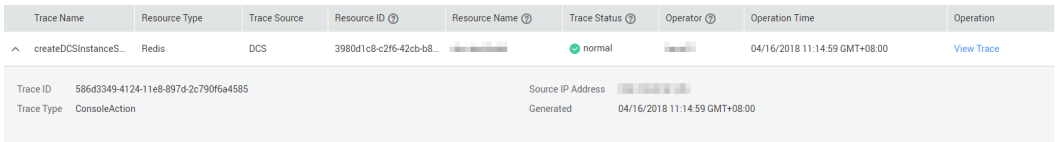
  When you select **Trace name**, you also need to select a specific trace name.

  When you select **Resource ID**, you also need to select a specific resource ID.

  When you select **Resource name**, you also need to select a specific resource name.
- **Operator**: Select a specific operator (a user other than tenant).
- **Trace Status**: Available options include **All trace status**, **normal**, **warning**, and **incident**. You can select only one of them.
- Start time and end time: You can specify the time period in which to query traces.

**Step 6** Click on the left of a trace to expand its details, as shown in **Figure 13-1**.

**Figure 13-1** Expanding trace details



**Step 7** Click **View Trace** in the **Operation** column. In the dialog box shown in **Figure 13-2**, the trace structure details are displayed.

**Figure 13-2** Viewing traces


```
{
    "time": "04/16/2018 11:14:59 GMT+08:00",
    "user": {
        "name": "[redacted]",
        "id": "5b64419de7f54010ace3ba9d63ece3c4",
        "domain": {
            "name": "[redacted]",
            "id": "cc9c0076188843ea938743dd166c7fef"
        }
    },
    "request": {
        "name": "[redacted]",
        "description": "",
        "engine": "Redis",
        "engine_version": "3.0.7",
        "capacity": 2,
        "password": "******",
        "vpc_id": "47c7ec3a-181f-4c3d-930f-de255c330f8b",
        "security_group_id": "2907f342-5960-4513-b8a4-1ba31eceabc9",
        "subnet_id": "cb6cbaf3-ebf0-4e62-8793-570d2a6de24b",
        "available_zones": [
            "ae04cf9d61544df3806a3feeb401b204"
        ],
        "product_id": "00301-31100-0--0",
        "no_password_access": false,
        "maintain_begin": "02:00:00",
```


**----End**