ALIBABA CLOUD

# Alibaba Cloud

## ApsaraDB for Redis

## Quick Start

Document Version: 20201124

Alibaba Cloud

# Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

# Document conventions

| Style | Description | Example |
| --- | --- | --- |
| Danger | A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results. | **Danger:** Resetting will result in the loss of user configuration data. |
| Warning | A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results. | **Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance. |
| Notice | A caution notice indicates warning information, supplementary instructions, and other content that the user must understand. | **Notice:** If the weight is set to 0, the server no longer receives new requests. |
| Note | A note indicates supplemental instructions, best practices, tips, and other content. | **Note:** You can use Ctrl + A to select all files. |
| > | Closing angle brackets are used to indicate a multi-level menu cascade. | Click **Settings**> **Network**> **Set network type**. |
| **Bold** | Bold formatting is used for buttons , menus, page names, and other UI elements. | Click **OK**. |
| `Courier font` | Courier font is used for commands | Run the `cd /d C:/window` command to enter the Windows system folder. |
| *Italic* | Italic formatting is used for parameters and variables. | `bae log list --instanceid` *Instance_ID* |
| [] or [a\|b] | This format is used for an optional value, where only one item can be selected. | `ipconfig [-all\|-t]` |
| {} or {a\|b} | This format is used for a required value, where only one item can be selected. | `switch {active\|stand}` |

# Table of Contents

# 1.Overview

This topic describes how to create an ApsaraDB for Redis instance and connect to databases on the instance. This topic also describes how to manage an instance.

## Intended audience

- Users who purchase an ApsaraDB for Redis instance for the first time.
- Users who require more information about how to connect to an ApsaraDB for Redis instance.

## Flowchart for an ApsaraDB for Redis instance

If you use the ApsaraDB for Redis instance for the first time, read Limits and ApsaraDB for Redis console.

ApsaraDB for Redis is fully compatible with native Redis commands. Alibaba Cloud provides some proprietary commands to improve user experience. For more information about supported commands and Alibaba Cloud proprietary commands, see Redis commands.

To purchase an ApsaraDB for Redis instance and use the instance, follow these steps:

Flowchart for an ApsaraDB for Redis instance

# 2.Step 1: Create an ApsaraDB for Redis instance

This topic describes how to create an ApsaraDB for Redis instance to meet your application requirements.

1. Log on to the ApsaraDB for Redis console.
2. On the **Instances** page, click **Create Instance**.
3. Select a billing method.
   - **Subscription**: You pay for each subscription instance when you create the instance. We recommend that you select this billing method if you have long-term requirements because it is more cost-effective than the pay-as-you-go billing. Higher discounts are offered for longer subscription periods.
   - **Pay-as-you-go**: You pay for only what you use. The instance is billed on an hourly basis. We recommend that you choose the pay-as-you-go billing for short-term needs. You can release your pay-as-you-go instance to save costs when you no longer need the instance.

   > **Notice**
   > - You can change the billing method only from pay-as-you-go to subscription.

4. Set the parameters in the following table.

| Parameter | Description |
| --- | --- |
| Region | The region where you want to create an ApsaraDB for Redis instance. You cannot change the region after you activate the instance.<br>- To maximize access speed, we recommend that you select a region that is close to the geographic location of your users.<br>- To enable connections over an internal network, make sure that the ApsaraDB for Redis instance is deployed in the same region as an Elastic Compute Service (ECS) instance. Otherwise, the instances can only communicate with each other over the Internet. Connections over the Internet may degrade the service performance. |
| Zone | The zone where the ApsaraDB for Redis instance is deployed. Each zone is an independent geographic location within a region. You can manage your workloads in different zones of the same region with identical service experience. |

| Parameter | Description |
| --- | --- |
| Network Type | ○ Classic Network: a traditional network type.<br>○ VPC: the recommended network type. A VPC network is an isolated virtual network that provides higher security and better performance than a classic network.<br>**Notice**<br>○ To enable connections over an internal network, make sure that the ApsaraDB for Redis instance is connected to the same type of network as an ECS instance or an ApsaraDB for RDS instance.<br>○ If only the VPC network type is used to enable connections, make sure that the ApsaraDB for Redis instance is connected to the same VPC network as an ECS instance or an ApsaraDB for RDS instance.<br>○ You can change the network type of ApsaraDB for Redis instances from Classic Network to VPC. However, you cannot change the network type from VPC to Classic Network. |
| VSwitch | A VSwitch is the basic network module used to build a VPC network. If no VSwitch is available in the VPC network, we recommend that you create a VSwitch. For more information, see Create a VSwitch. |
| Edition | ○ Community Edition: This edition is compatible with the Redis protocol and provides database solutions that support hybrid storage in memory and disks.<br>○ Enhanced Edition: This edition is developed on the basis of ApsaraDB for Redis Community Edition and has been optimized in terms of performance, storage, and data schemas. For more information, see Overview. |
| Series | Enhanced Edition includes the following series:<br>○ Performance-enhanced series. For more information, see Performance-enhanced instances and Integration of multiple Redis modules.<br>○ Hybrid-storage series. For more information, see Hybrid-storage instances.<br>**Note** Community Edition does not support these series. |

| Parameter | Description |
| --- | --- |
| Version | ApsaraDB for Redis supports the following engine versions:<br>○ 2.8<br>○ 4.0<br>○ 5.0<br>**Note**<br>○ If the required database version does not appear, we recommend that you specify a different database edition or instance type.<br>○ The ApsaraDB for Redis instances with Redis 2.8 will soon be phased out. We recommend that you create an instance with the latest engine version to use more features and achieve stable performance. |
| Architecture Type | ○ Cluster<br>○ Standard<br>○ Read-write Splitting<br>For more information, see Overview. |
| Shards | The number of shards on a cluster instance.<br>**Note** You can specify the number of shards only on cluster instances. |
| Node Type | ○ If the architecture type is set to Standard or Cluster, the node type can only be set to Master-Replica. A master-replica node runs in a hot standby mode and supports data persistence.<br>○ If the architecture type is set to Read-Write Splitting, the node type can be set to:<br>■ 1 Read-only Replica<br>■ 3 Read-only Replicas<br>■ 5 Read-only Replicas |

| Parameter | Description |
| --- | --- |
| Instance Class | Each instance type contains a group of configurations, such as the memory capacity, maximum number of concurrent connections, and maximum internal bandwidth. For more information, see Overview.<br><br>**Note** After you create an ApsaraDB for Redis instance, metadata for a database is generated and uses a minimum amount of storage space.<br>○ On an ApsaraDB for Redis instance of the standard architecture type, the size of metadata ranges from 30 MB to 50 MB.<br>○ On each shard of an ApsaraDB for Redis instance of the cluster architecture type, the size of metadata ranges from 30 MB to 50 MB. In a cluster, the total amount of storage space that metadata uses equals the total size of metadata on each shard. |

5. Set a password.
   - **Now**: In the **Password** field, enter a password.
   - **Later**: You can set a password after you create the instance. For more information, see Change the password.
6. Set **Instance Name** and **Quantity**. If you purchase a subscription instance, you must set **Duration**.
7. Click **Buy Now**.
8. On the **Confirm Order** page that appears, click **ApsaraDB for KVStore (Subscription) Agreement of Service**.
9. Read and select the ApsaraDB for KVStore (Subscription) Agreement of Service checkbox.
10. Click **Pay**.

> **Note** After you pay for the order, the system prompts that you have activated the instance. Then, wait up to five minutes to check the instance displayed in the console.

## FAQ

- Q: How long does it take for the system to create an instance?

  A: The time it takes to create an instance depends on the number of data shards on the instance. A larger number of data shards require more resources. As a result, it takes more time to allocate resources. For example, it takes 2 to 3 minutes to create a standard master-replica instance, 10 to 15 minutes to create a 128-shard cluster master-replica instance, and 20 to 40 minutes to create a 256-shard cluster master-replica instance.

  > **Note**
  > - For standard instances, the amount of time depends on single-shard instances.
  > - The primary node of a read/write splitting instance processes read and write requests. Each primary node is regarded as a shard, and each read replica is regarded as a shard.

- Q: Why did I fail to create a standalone instance of ApsaraDB for Redis?

A: Starting December 19, 2019, standalone instances of ApsaraDB for Redis are phased out. For more information, see Deprecated standalone instances of ApsaraDB for Redis.

## Related API operations

| API | Description |
| --- | --- |
| CreateInstance | Creates an ApsaraDB for Redis instance. |

# 3.Step 2: Set IP address whitelists

Before you use an ApsaraDB for Redis instance, you must set one or more IP address whitelists or specify Elastic Compute Service (ECS) security groups as whitelists. This ensures data security and guarantee high performance of ApsaraDB for Redis. You can add client IP addresses or Classless Inter-Domain Routing (CIDR) blocks to a whitelist. We recommend that you update your whitelists on a regular basis to improve data security in ApsaraDB for Redis.

## Prerequisites

- The ApsaraDB for Redis instance is upgraded to the latest minor version.

  > **Note** If your instance is not upgraded to the latest minor version, a message may appear when you set a whitelist. In this case, you can upgrade the instance to the latest minor version. For more information, see Upgrade the minor version.

- To specify an ECS security group as a whitelist, the engine version of the instance must be Redis 4.0 or later.

## Notes

When you use Data Management Service (DMS) to log on to the ApsaraDB for Redis instance, the system automatically creates a whitelist named ali_dms_group. Do not manually add other IP addresses to this whitelist. Otherwise, IP address loss may occur due to the DMS changes.

## Set one or more whitelists

> **Note** You can set one or more IP whitelists and specify ECS security groups as whitelists of an ApsaraDB for Redis instance. Both IP addresses in IP whitelists and ECS instances in security groups are allowed to access the instance.

1. Log on to the ApsaraDB for Redis console.
2. On the top of the page, select the region where the instance is deployed.
3. On the **Instances** page, click the instance ID of the target instance or choose More > **Manage** in the **Actions** column for the instance.
4. In the left-side navigation pane of the **Instance Information** page, click **Whitelist Settings**.
5. On the **Whitelist Settings** page, use the following methods to manage whitelists:

   > **danger** When you use DMS to log on to the ApsaraDB for Redis instance, the system automatically creates a whitelist named ali_dms_group. Do not manually add other IP addresses to this whitelist. Otherwise, IP address loss may occur due to the DMS changes.

   - Create a whitelist with a custom whitelist name:
     a. Click **Add Whitelist**.

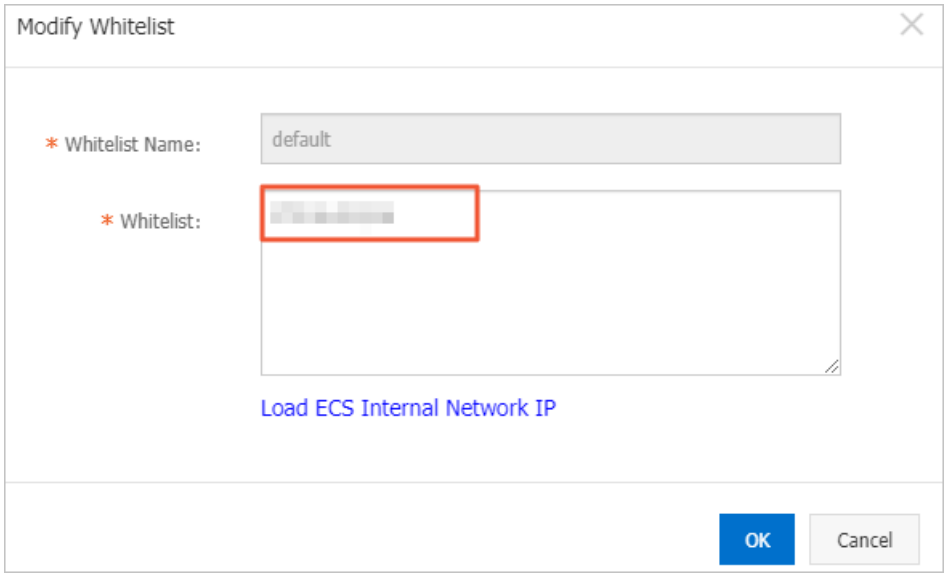
b. In the Add Whitelist dialog box, set **Whitelist Name**.

> **Note**
>
> - The name of the whitelist must be 2 to 32 characters in length, and can contain lowercase letters, digits, and underscores (_). It must start with a lowercase letter and end with a letter or digit.
> - After you create a whitelist, you cannot change the name of the whitelist.

- If you want to use an existing whitelist, click **Modify** on the right of the whitelist.

6. In the **Add Whitelist** or **Modify Whitelist** dialog box, perform one of the following steps:
   - Manually modify the value of **Whitelist**:
     a. In the **Whitelist** field, enter the IP addresses or CIDR blocks that can be used to connect to the ApsaraDB for Redis instance.

Manually modify the whitelist

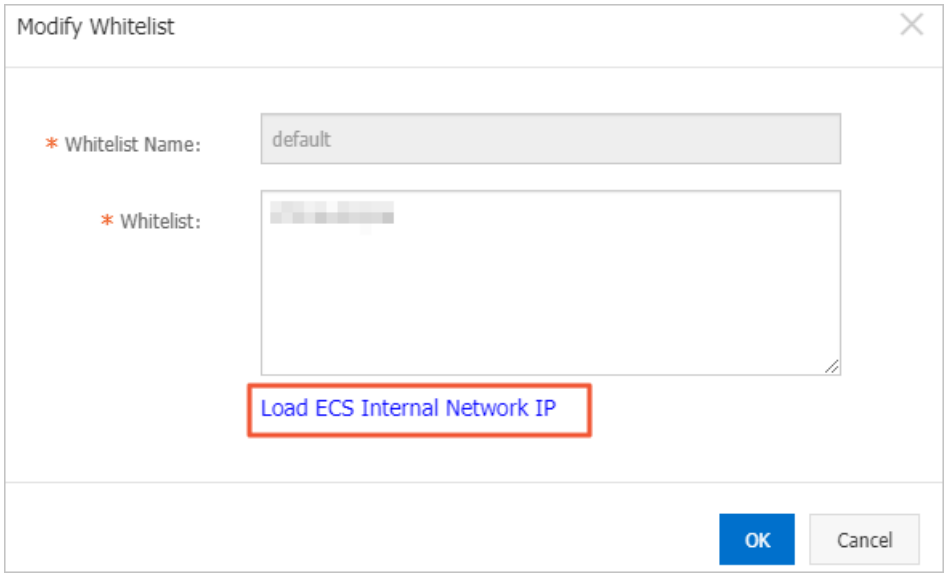



> **Note**
>
> - If you set the whitelist to `0.0.0.0/0`, you allow connections from all IP addresses. To ensure data security, we recommended that you do not specify this CIDR block.
> - If you set the whitelist to a CIDR block, such as `10.10.10.0/24`, you allow connections from all the IP addresses within the CIDR block.
> - Separate multiple IP addresses or CIDR blocks with commas (,) and do not leave spaces before or after each comma.
> - You can add up to 1,000 IP addresses or CIDR blocks to each whitelist.

     b. Click **OK**.
   - Load internal IP addresses of ECS instances under your Alibaba Cloud account:

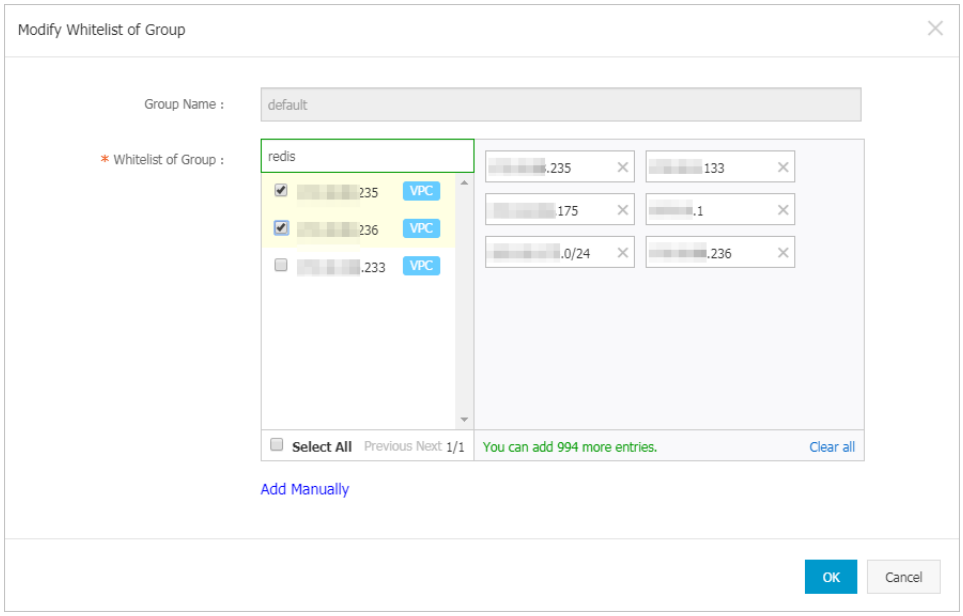
a. Click **Load ECS Internal IP Addresses**.

Load ECS internal IP addresses




b. Select internal IP addresses of ECS instances.

Select internal IP addresses of ECS instances



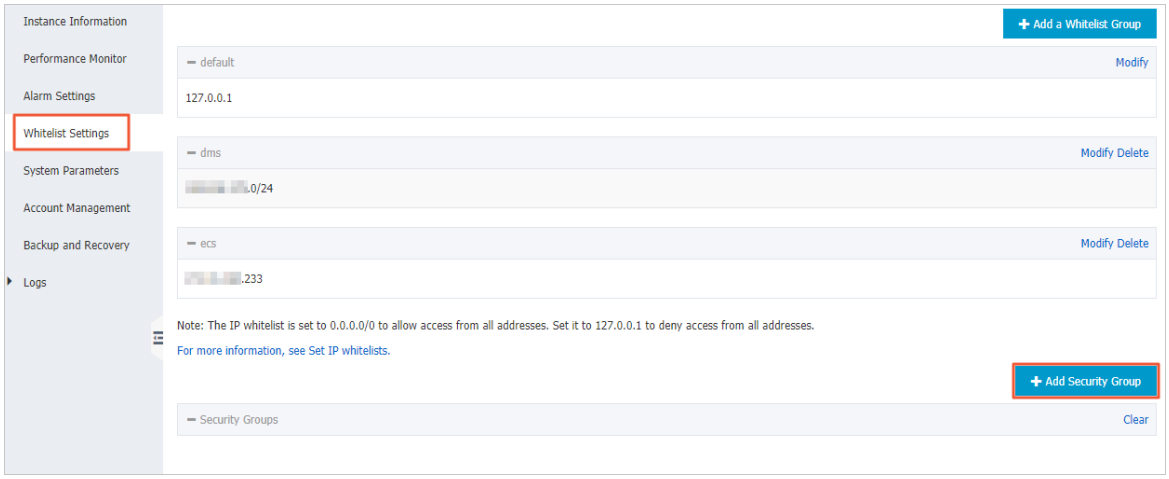

> **Note** You can search ECS instances by instance name, ID, or IP address in the search bar above the list of ECS internal IP addresses.

c. Click **OK**.

## Specify ECS security groups as whitelists

A security group serves as a virtual firewall to limit the inbound and outbound network traffic of ECS instances that belong to the security group. After you specify a security group as a whitelist of the ApsaraDB for Redis instance, all ECS instances in the security group are allowed to access the instance. For more information about ECS security groups, see Security group overview.

1. Log on to the ApsaraDB for Redis console.
2. On the top of the page, select the region where the instance is deployed.
3. On the **Instances** page, click the instance ID of the target instance or choose More > **Manage** in the **Actions** column for the instance.
4. In the left-side navigation pane of the **Instance Information** page, click **Whitelist Settings**.
5. On the **Whitelist Settings** page, click **Add Security Group** to add an ECS security group.




6. In the **Add Security Group** dialog box, perform the following steps:

> **Note** You can add up to 10 ECS security groups for each ApsaraDB for Redis instance.

   i. Select one or more security groups that you want to add.
   ii. Click OK.

## Related operations

| API | Description |
| --- | --- |
| DescribeSecurityIps | Queries IP address whitelists of an ApsaraDB for Redis instance. |
| ModifySecurityIps | Modifies IP address whitelists of an ApsaraDB for Redis instance. |

# 4.Step 3: Connect to the instance

## 4.1. Use DMS

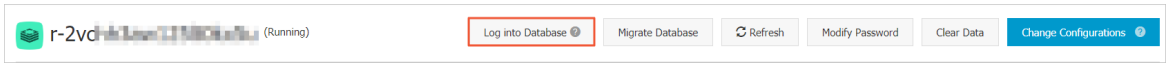
Data Management Service (DMS) is a web service designed to access and manage data in the cloud. The service supports data sources such as Redis, MySQL, SQL Server, PostgreSQL, and MongoDB. DMS provides the following features: data management, object management, data transfer, and instance management.

### Limits

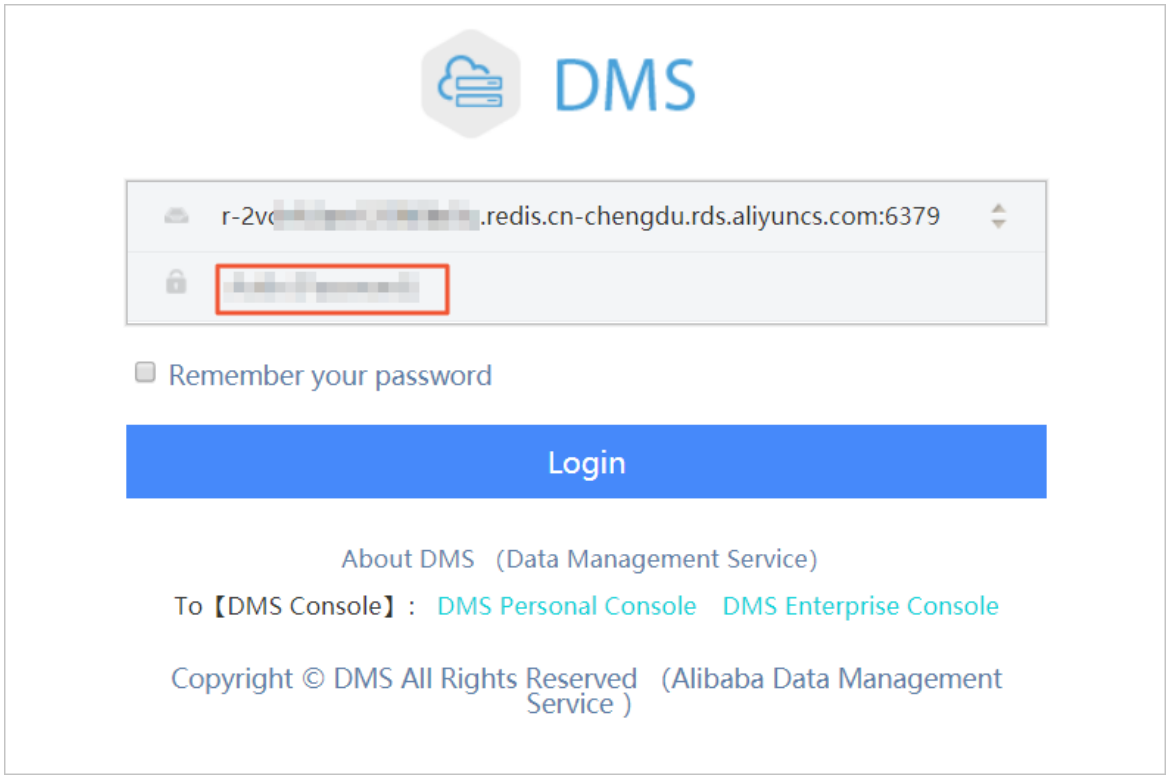
- Connecting to the ApsaraBD for Redis instances without using proxy servers or over the Internet is not supported in the DMS console.
- When you use DMS to log on to a performance-enhanced instance of ApsaraDB for Redis Enhanced Edition, you cannot use the commands described in Commands supported by performance-enhanced instances.

### Log on to the ApsaraDB for Redis console

1. Log on to the ApsaraDB for Redis console.
2. On the top of the page, select the region where the ApsaraDB for Redis instance is deployed.
3. In the left-side navigation pane, click **Instances**. On the Instances page that appears, find the instance that you want to manage. Click the instance ID, or select > **Manage** in the **Actions** column for the instance.
4. On the **Instance Information** page, click **Log into Database**.



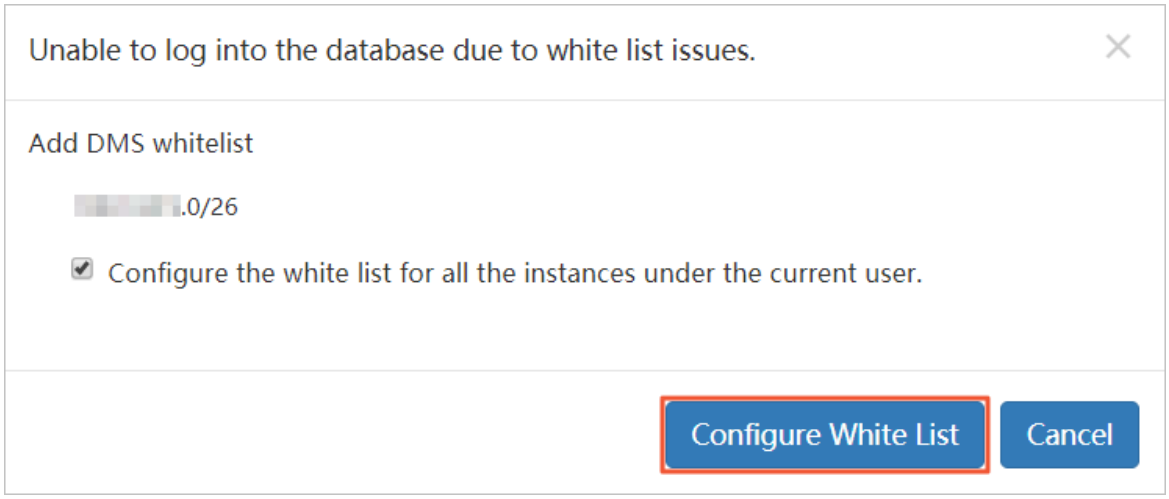

5. On the DMS logon page, enter the instance password.




> **Note**
>
> - If the network type of the ApsaraDB for Redis instance is VPC and password-free access is enabled, you do not need to enter the password.
> - Instances in both classic and VPC networks support DMS access. When you use DMS to log on to an instance in a VPC network, the system requests a special channel. Therefore, you need to wait a moment before logging on to this instance for the first time.
> - Whenever you log on to the DMS console, ApsaraDB for Redis runs the SCAN command to scan data and delete some expired data in ApsaraDB for Redis.

6. Click **Login**.
7. In the **Unable to log into the database due to whitelist issues** dialog box, click **Configure Whitelist**.

> ⚠ **danger** The system creates a group named ali_dms_group in the Whitelist of the ApsaraDB for Redis instance and adds the IP address of the DMS client to the group. Do not manually add other IP addresses to this group. Otherwise, IP address loss may occur due to DMS changes.

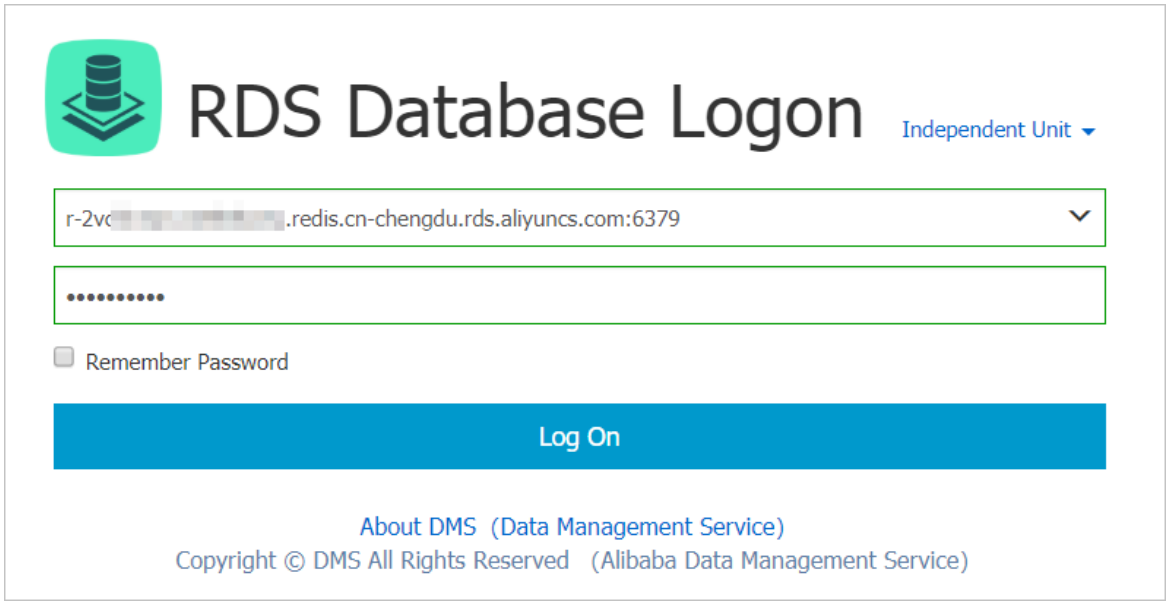



**Note** This dialog box appears only if you have never logged on to the ApsaraDB for Redis instance before by using the DMS console.

8. Click **Login**.

## Log on to the DMS console

1. Log on to the DMS console.
2. Enter the connection address and password of the instance that you want to manage.



> **Note**
>
> - If the network type of the ApsaraDB for Redis instance is VPC and password-free access is enabled, you do not need to enter the password.
> - Instances in both classic and VPC network support DMS. When you use the DMS console to log on to an instance in a VPC network, the system requests a special channel. Therefore, you need to wait a moment before logging on to this instance for the first time.
> - Whenever you log on to DMS, ApsaraDB for Redis runs the SCAN command to scan data and delete some expired data in ApsaraDB for Redis.

3. - If you have used DMS to log on to the ApsaraDB for Redis instance before, click **Log On**.
   - If you have never used DMS to log on to the ApsaraDB for Redis instance before, follow these steps.
     a. In the **The DMS server is not on the RDS whitelist** dialog box, copy the IP address of the DMS sever and click **Close**.
     b. Add the IP address of the DMS client to a whitelist of the ApsaraDB for Redis instance. For more information, see Add Whitelist.
     c. Log on to the DMS console.
     d. Enter the connection string and password of the instance that you want to manage.
     e. Click **Log On**.

## FAQ

- How do I use a custom account to log on to DMS?
- If you use a custom account to connect to the ApsaraDB for Redis instance, the connection password must be in the format of `<user>:<password>`. For example, if the username of a custom account is `admin` and the password is `Fakepass666`, the password used to connect to the ApsaraDB for Redis instance must be in the format of `admin:Fakepass666`.
- What can I do if I fail to connect to an ApsaraDB for Redis instance?
- If a connection issue occurs, resolve the issue by following the instructions in Troubleshoot connection issues in ApsaraDB for Redis. For more information about DMS, see DMS Documentation.

# 4.2. Use a Redis client

You can connect to an ApsaraDB for Redis instance by using clients for different programming languages.

## Prerequisites

- The internal IP address of the Elastic Compute Service (ECS) instance or the public IP address of the on-premises machine is added to an IP address whitelist of the ApsaraDB for Redis instance. For more information, see Set IP address whitelists.
- If you use a custom account to connect to the ApsaraDB for Redis instance, the connection string must be in the format of `<user>:<password>`. For example, if the username of the custom account is `admin` and the password is `Rp829dlwa`, you can use `admin:Rp829dlwa` to log on to the instance.

## Considerations

- By default, cluster instances use the proxy mode. In this mode, you can access ApsaraDB for Redis instances by using the endpoint of the proxy server in the same way as you access standard instances of ApsaraDB for Redis.

  > **Note** If you use a private endpoint to connect to an ApsaraDB for Redis instance, you can connect to the instance in the same way as you connect to an open source Redis Cluster.

- If you enable password-free access for instances in the same Virtual Private Cloud (VPC) network, you can connect to databases of the ApsaraDB for Redis instance without passwords.
- For more information about connection issues, see Troubleshooting for connection issues in ApsaraDB for Redis.

## Redis client

The database service of ApsaraDB for Redis is compatible with that of native Redis. Therefore, you can connect to both database services in similar ways. All clients that are compatible with the Redis protocol support connections to ApsaraDB for Redis. You can use any of these clients that are suitable for your applications.

For more information about Redis clients, visit Clients.

- Jedis client
- Lettuce client (Not recommended)
- PhpRedis client
- Redis-py client
- C or C++ client
- .NET client
- node-redis client
- C# client StackExchange.Redis

## Jedis client

You can use a Jedis client to connect to ApsaraDB for Redis in the following ways:

- Single Jedis connection. This method is not recommended because a client cannot automatically reconnect to ApsaraDB for Redis after a connection times out.
- JedisPool-based connection (Recommended).

To use a Jedis client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and install the Jedis client. For more information, see Jedis.
2. Example of single Jedis connection

i. Launch the Eclipse client, create a project, and then enter the following code:

```
import redis.clients.jedis.Jedis;
public class jedistest {
public static void main(String[] args) {
try {
   String host = "xx.kvstore.aliyuncs.com";//You can find the connection address in the console.
   int port = 6379;
   Jedis jedis = new Jedis(host, port);
   //Authentication information.
   jedis.auth("password");//password
   String key = "redis";
   String value = "aliyun-redis";
   //Select a database. Default value: 0.
   jedis.select(1);
   //Set a key.
   jedis.set(key, value);
   System.out.println("Set Key " + key + " Value: " + value);
   //Obtain the configured key and value.
   String getvalue = jedis.get(key);
   System.out.println("Get Key " + key + " ReturnValue: " + getvalue);
   jedis.quit();
   jedis.close();
}
catch (Exception e) {
 e.printStackTrace();
 }
}
}
```

ii. Run the project. If you see the following result in Eclipse, you have connected to ApsaraDB for Redis.

```
Set Key redis Value aliyun-redis
Get Key redis ReturnValue aliyun-redis
```

Then, you can use a Jedis client to manage your ApsaraDB for Redis instance. You can also connect to your ApsaraDB for Redis instance by using JedisPool.

3. Example of JedisPool-based connection

i. Launch the Eclipse client, create a project, and then configure the following pom file:

```
<dependency>
<groupId>redis.clients</groupId>
<artifactId>jedis</artifactId>
<version>2.7.2</version>
<type>jar</type>
<scope>compile</scope>
</dependency>
```

ii. Add the following application to the project:

```
import org.apache.commons.pool2.PooledObject;
import org.apache.commons.pool2.PooledObjectFactory;
import org.apache.commons.pool2.impl.DefaultPooledObject;
import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisPool;
import redis.clients.jedis.JedisPoolConfig;
```

iii. If your Jedis client version is Jedis-2.7.2, enter the following code in the project:

```
JedisPoolConfig config = new JedisPoolConfig();
//The maximum number of idle connections. You can customize this parameter. The value cannot e
xceed the maximum number of connections that is supported by each ApsaraDB for Redis instance.
config.setMaxIdle(200);
//The maximum number of connections. You can customize this parameter. The value cannot excee
d the maximum number of connections that is supported by each ApsaraDB for Redis instance.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
JedisPool pool = new JedisPool(config, host, 6379, 3000, password);
Jedis jedis = null;
try {
jedis = pool.getResource();
/// ... do stuff here ... for example
jedis.set("foo", "bar");
String foobar = jedis.get("foo");
jedis.zadd("sose", 0, "car");
jedis.zadd("sose", 0, "bike");
Set<String> sose = jedis.zrange("sose", 0, -1);
} finally {
if (jedis ! = null) {
jedis.close();
}
}
/// ... when closing your application:
pool.destroy();
```

iv. If your Jedis client version is Jedis-2.6 or Jedis-2.5, enter the following code in the project:

```
JedisPoolConfig config = new JedisPoolConfig();
//The maximum number of idle connections. You can customize this parameter. The value cannot e
xceed the maximum number of connections that is supported by each ApsaraDB for Redis instance.
config.setMaxIdle(200);
//The maximum number of connections. You can customize this parameter. The value cannot excee
d the maximum number of connections that is supported by each ApsaraDB for Redis instance.
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "Password";
JedisPool pool = new JedisPool(config, host, 6379, 3000, password);
Jedis jedis = null;
boolean broken = false;
try {
   jedis = pool.getResource();
   /// ... do stuff here ... for example
   jedis.set("foo", "bar");
   String foobar = jedis.get("foo");
   jedis.zadd("sose", 0, "car");
   jedis.zadd("sose", 0, "bike");
   Set<String> sose = jedis.zrange("sose", 0, -1);
}
catch(Exception e)
{
   broken = true;
} finally {
if (broken) {
   pool.returnBrokenResource(jedis);
} else if (jedis ! = null) {
   pool.returnResource(jedis);
 }
}
```

v. Run the project. If you see the following result in Eclipse, you have connected to ApsaraDB for Redis.

```
Set Key redis Value aliyun-redis
Get Key redis ReturnValue aliyun-redis
```

Then, you can use a Jedis client to manage your ApsaraDB for Redis instance.

## Lettuce client

A Lettuce client supports synchronous and asynchronous communication based on comprehensive Redis API operations. A Lettuce client does not automatically reconnect to an instance after multiple requests time out. If failures occur in ApsaraDB for Redis and cause failovers for proxy servers or database nodes, connection timeout may occur. This may result in the failure to reconnect to ApsaraDB for Redis. To avoid such risks, we recommend that you use Jedis clients. For more information, see Jedis client.

For more information, see Lettuce.

## TairJedis client

TairJedis is an ApsaraDB for Redis client developed by Alibaba Cloud. TairJedis supports the features of Jedis and features dedicated for ApsaraDB for Redis Enhanced Edition (Tair). For example, TairJedis supports the Tair structures and Tair commands.

For more information, see TairJedis on GitHub.

## PhpRedis client

To use a PhpRedis client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and install the PhpRedis client. For more information, see PhpRedis.
2. In an editor that supports PHP editing, enter the following code:

```
<? php
/* Replace the following parameter values with the host name and port number of the instance. */
$host = "localhost";
$port = 6379;
/* Replace the following parameter values with the ID and password of the instance. */
$user = "test_username";
$pwd = "test_password";
$redis = new Redis();
if ($redis->connect($host, $port) == false) {
    die($redis->getLastError());
 }
if ($redis->auth($pwd) == false) {
    die($redis->getLastError());
}
 /* You can manage the database after authentication. For more information, visit https://github.com/p
hpRedis/phpredis. */.
if ($redis->set("foo", "bar") == false) {
    die($redis->getLastError());
}
$value = $redis->get("foo");
echo $value;
? >
```

3. Run the code. Then, you can use a PhpRedis client to connect to your ApsaraDB for Redis instance. For more information, visit phpredis.

## Redis-py client

To use a redis-py client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and install the redis-py client. For more information, see redis-py.
2. In an editor that supports Python, enter the following code. You can use a redis-py client to connect to the ApsaraDB for Redis instance and manage the database.

```
#! /usr/bin/env python
#-*- coding: utf-8 -*-
import redis
#Replace the following parameter values with the host name and port number of the instance.
host = 'localhost'
port = 6379
#Replace the following parameter value with the password of the instance.
pwd = 'test_password'
r = redis.StrictRedis(host=host, port=port, password=pwd)
#You can perform database operations after you establish a connection. For more information, visit https://
github.com/andymccurdy/redis-py.
r.set('foo', 'bar');
print r.get('foo')
```

## C or C++ client

To use a C or C++ client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download, compile, and install the C client by using the following code:

```
git clone https://github.com/redis/hiredis.git
cd hiredis
make
sudo make install
```

2. Enter the following code in the C or C++ editor:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
unsigned int j;
redisContext *c;
redisReply *reply;
if (argc < 4) {
```

```
        printf("Usage: example xxx.kvstore.aliyuncs.com 6379 instance_id password\n");
        exit(0);
}
const char *hostname = argv[1];
const int port = atoi(argv[2]);
const char *instance_id = argv[3];
const char *password = argv[4];
struct timeval timeout = { 1, 500000 }; // 1.5 seconds
c = redisConnectWithTimeout(hostname, port, timeout);
if (c == NULL || c->err) {
if (c) {
        printf("Connection error: %s\n", c->errstr);
        redisFree(c);
} else {
        printf("Connection error: can't allocate redis context\n");
}
exit(1);
}
/* AUTH */
reply = redisCommand(c, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);
/* PING server */
reply = redisCommand(c,"PING");
printf("PING: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key */
reply = redisCommand(c,"SET %s %s", "foo", "hello world");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);
/* Set a key using binary safe API */
reply = redisCommand(c,"SET %b %b", "bar", (size_t) 3, "hello", (size_t) 5);
printf("SET (binary API): %s\n", reply->str);
freeReplyObject(reply);
/* Try a GET and two INCR */
reply = redisCommand(c,"GET foo");
printf("GET foo: %s\n", reply->str);
freeReplyObject(reply);
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
```

```
freeReplyObject(reply);
/* again ... */
reply = redisCommand(c,"INCR counter");
printf("INCR counter: %lld\n", reply->integer);
freeReplyObject(reply);
/* Create a list of numbers, from 0 to 9 */
reply = redisCommand(c,"DEL mylist");
freeReplyObject(reply);
for (j = 0; j < 10; j++) {
    char buf[64];
    snprintf(buf,64,"%d",j);
    reply = redisCommand(c,"LPUSH mylist element-%s", buf);
    freeReplyObject(reply);
  }
/* Let's check what we have inside the list */
reply = redisCommand(c,"LRANGE mylist 0 -1");
if (reply->type == REDIS_REPLY_ARRAY) {
    for (j = 0; j < reply->elements; j++) {
    printf("%u) %s\n", j, reply->element[j]->str);
}
}
freeReplyObject(reply);
/* Disconnects and frees the context */
redisFree(c);
return 0;
}
```

3. Compile the code.

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

4. Run a test.

```
example xxx.kvstore.aliyuncs.com 6379 instance_id password
```

Now, the C or C++ client is connected to the ApsaraDB for Redis instance.

## .NET client

To use a .NET client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and use the .NET client.

```
git clone https://github.com/ServiceStack/ServiceStack.Redis
```

2. Create a .NET project on the .NET client.

3. Add the reference file stored in the library file directory ServiceStack.Redis/lib/tests to the client.
4. Enter the following code in the .NET project to connect to the ApsaraDB for Redis instance. For more information about API operations, visit ServiceStack.Redis.

```
using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Text;
 using System.Threading.Tasks;
 using ServiceStack.Redis;
 namespace ServiceStack.Redis.Tests
 {
     class Program
 {
 public static void RedisClientTest()
 {
     string host = "127.0.0.1";/*The IP address of the host.*/
     string password = "password";/*Password*/
     RedisClient redisClient = new RedisClient(host, 6379, password);
     string key = "test-aliyun";
     string value = "test-aliyun-value";
     redisClient.Set(key, value);
     string listKey = "test-aliyun-list";
     System.Console.WriteLine("set key " + key + " value " + value);
     string getValue = System.Text.Encoding.Default.GetString(redisClient.Get(key));
     System.Console.WriteLine("get key " + getValue);
     System.Console.Read();
 }
 public static void RedisPoolClientTest()
 {
     string[] testReadWriteHosts = new[] {
     "redis://password@127.0.0.1:6379"/*redis://Password@IP address:Port*/
 };
 RedisConfig.VerifyMasterConnections = false;//Required.
 PooledRedisClientManager redisPoolManager = new PooledRedisClientManager(10/*Number of connec
tion pools*/, 10/*Connection pool timeout value*/, testReadWriteHosts);
 for (int i = 0; i < 100; i++){
     IRedisClient redisClient = redisPoolManager.GetClient();//Obtain the connection.
     RedisNativeClient redisNativeClient = (RedisNativeClient)redisClient;
     redisNativeClient.Client = null;//ApsaraDB for Redis does not support the CLIENT SETNAME comman
d. Set Client to null.
```

```
try
{
    string key = "test-aliyun1111";
    string value = "test-aliyun-value1111";
    redisClient.Set(key, value);
    string listKey = "test-aliyun-list";
    redisClient.AddItemToList(listKey, value);
    System.Console.WriteLine("set key " + key + " value " + value);
    string getValue = redisClient.GetValue(key);
    System.Console.WriteLine("get key " + getValue);
    redisClient.Dispose();//
}catch (Exception e)
{
    System.Console.WriteLine(e.Message);
}
}
    System.Console.Read();
}
static void Main(string[] args)
{
    //Single-connection mode.
    RedisClientTest();
    //Connection-pool mode.
    RedisPoolClientTest();
}
}
}
```

## node-redis client

To use a node-redis client to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and install a node-redis client.

   ```
   npm install hiredis redis
   ```

2. Enter and run the following code on the node-redis client to connect to the ApsaraDB for Redis instance.

   ```
   var redis = require("redis"),
   client = redis.createClient(<port>, <"host">, {detect_buffers: true});
   client.auth("password", redis.print)
   ```

> **Note** The port field specifies the port number of the ApsaraDB for Redis instance. Default value: 6379. The host field specifies the endpoint of the ApsaraDB for Redis instance. For example:

```
client = redis.createClient(6379, "r-abcdefg.redis.rds.aliyuncs.com", {detect_buffers: true});
```

3. Use the ApsaraDB for Redis instance.

```
// Write data to the instance.
client.set("key", "OK");
// Query data on the instance. The data of the String type is returned.
client.get("key", function (err, reply) {
console.log(reply.toString()); // print `OK`
});
// If the input parameter is a buffer, the returned value is also a buffer.
client.get(new Buffer("key"), function (err, reply) {
console.log(reply.toString()); // print `<Buffer 4f 4b>`
});
client.quit();
```

## C# client StackExchange.Redis

To use the C# client StackExchange.Redis to connect to an ApsaraDB for Redis instance, perform the following steps:

1. Download and install StackExchange. Redis.
2. Add StackExchange.Redis

```
using StackExchange.Redis;
```

3. Initialize ConnectionMultiplexer.

   ConnectionMultiplexer is the core of StackExchange.Redis, and shared and reused in the entire application. You must use ConnectionMultiplexer as a singleton. ConnectionMultiplexer is initialized in the following way:

```
// redis config
private static ConfigurationOptions configurationOptions = ConfigurationOptions.Parse("127.0.0.1:63
79,password=xxx,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(configurationOptions);
      }
    }
  }
  return redisConn;
}
```

> **Note**
>
> ConfigurationOptions contains multiple options, such as keepAlive, connectRetry, and name. For more information, see StackExchange.Redis.ConfigurationOptions.

4. The lightweight object returned by GetDatabase(), which does not need to be stored. You can obtain the object from the object of ConnectionMultiplexer each time.

```
redisConn = getRedisConn();
var db = redisConn.GetDatabase();
```

5. The following examples show multiple types of data structures, which are strings, hashes, lists, sets, and sorted sets. The API operations used in these examples are different from their usage in the native Redis service.
   - string

```
//set get
string strKey = "hello";
string strValue = "world";
bool setResult = db.StringSet(strKey, strValue);
Console.WriteLine("set " + strKey + " " + strValue + ", result is " + setResult);
//incr
string counterKey = "counter";
long counterValue = db.StringIncrement(counterKey);
Console.WriteLine("incr " + counterKey + ", result is " + counterValue);
//expire
db.KeyExpire(strKey, new TimeSpan(0, 0, 5));
Thread.Sleep(5 * 1000);
Console.WriteLine("expire " + strKey + ", after 5 seconds, value is " + db.StringGet(strKey));
//mset mget
KeyValuePair<RedisKey, RedisValue> kv1 = new KeyValuePair<RedisKey, RedisValue>("key1", "value1
");
KeyValuePair<RedisKey, RedisValue> kv2 = new KeyValuePair<RedisKey, RedisValue>("key2", "value2
");
db.StringSet(new KeyValuePair<RedisKey, RedisValue>[] {kv1,kv2});
RedisValue[] values = db.StringGet(new RedisKey[] {kv1.Key, kv2.Key});
Console.WriteLine("mget " + kv1.Key.ToString() + " " + kv2.Key.ToString() + ", result is " + values[0] + "
&&" + values[1]);
```

- hash

```
string hashKey = "myhash";
//hset
db.HashSet(hashKey,"f1","v1");
db.HashSet(hashKey,"f2", "v2");
HashEntry[] values = db.HashGetAll(hashKey);
//hgetall
Console.Write("hgetall " + hashKey + ", result is");
for (int i = 0; i < values.Length;i++)
{
 HashEntry hashEntry = values[i];
 Console.Write(" " + hashEntry.Name.ToString() + " " + hashEntry.Value.ToString());
}
Console.WriteLine();
```

- list

```
//list key
string listKey = "myList";
//rpush
db.ListRightPush(listKey, "a");
db.ListRightPush(listKey, "b");
db.ListRightPush(listKey, "c");
//lrange
RedisValue[] values = db.ListRange(listKey, 0, -1);
Console.Write("lrange " + listKey + " 0 -1, result is ");
for (int i = 0; i < values.Length; i++)
{
 Console.Write(values[i] + " ");
}
Console.WriteLine();
```

- set

```
//set key
string setKey = "mySet";
//sadd
db.SetAdd(setKey, "a");
db.SetAdd(setKey, "b");
db.SetAdd(setKey, "c");
//sismember
bool isContains = db.SetContains(setKey, "a");
Console.WriteLine("set " + setKey + " contains a is " + isContains );
```

- sorted set

```
string sortedSetKey = "myZset";
//sadd
db.SortedSetAdd(sortedSetKey, "xiaoming", 85);
db.SortedSetAdd(sortedSetKey, "xiaohong", 100);
db.SortedSetAdd(sortedSetKey, "xiaofei", 62);
db.SortedSetAdd(sortedSetKey, "xiaotang", 73);
//zrevrangebyscore
RedisValue[] names = db.SortedSetRangeByRank(sortedSetKey, 0, 2, Order.Ascending);
Console.Write("zrevrangebyscore " + sortedSetKey + " 0 2, result is ");
for (int i = 0; i < names.Length; i++)
{
 Console.Write(names[i] + " ");
}
Console.WriteLine();
```

# 4.3. Use redis-cli to connect to ApsaraDB for Redis

This topic describes how to use the Redis command-line tool (redis-cli) to connect to an ApsaraDB for Redis instance.

## Introduction to redis-cli

Redis-cli is a built-in command-line tool of Redis. You can use redis-cli to connect to an ApsaraDB for Redis instance and manage data.

Redis-cli allows you to connect to an ApsaraDB for Redis instance from the Linux system deployed on an Elastic Compute Service (ECS) instance, or access an ApsaraDB for Redis instance from a local host over the Internet. ApsaraDB for Redis allows you to access an instance over a private network to ensure high security and performance. If you want to connect an ECS instance to an ApsaraDB for Redis instance deployed in the same Virtual Private Cloud (VPC) network or to an ApsaraDB for Redis instance deployed in a classic network in the same region, you can use redis-cli on the ECS instance. You can also connect to an ApsaraDB for Redis instance from your on-premises host over the Internet. To do so, follow the instructions in Use a public endpoint to apply for a public endpoint. Then, follow the instructions in the Connect to an ApsaraDB for Redis instance section of this topic to connect to the instance.

## Install redis-cli

Install a Linux-based version of Redis to use redis-cli. For more information, visit the Redis official website.

## Prerequisites

Connections over a private network

- If the network type for both the ECS instance and the ApsaraDB for Redis instance is VPC, these two instances must be deployed in the same VPC network of a region.

- If the network type for both the ECS instance and the ApsaraDB for Redis instance is classic network, these two instances must be deployed in the same region.
- The private IP address of the ECS instance is added to the whitelist of the ApsaraDB for Redis instance.
- The Linux-based version of Redis is installed on the ECS instance.
- If you use a custom account to connect to the ApsaraDB for Redis instance, the format of the connection string must be `<user>:<password>`. For example, if the username of the custom account is `admin` and the password is `Rp829dlwa`, you can use `admin:Rp829dlwa` to log on to the instance.

Connections over the Internet

- You have applied for a public endpoint for the ApsaraDB for Redis instance. For more information, see Use a public endpoint.
- You have added the public IP address of the local host to the whitelist of the ApsaraDB for Redis instance.
- The operating system of the local host must be Linux.
- You have installed the Linux-based version of Redis on the local host.
- If you use a custom account to connect to the ApsaraDB for Redis instance, the format of the connection string must be `<user>:<password>`. For example, if the username of the custom account is `admin` and the password is `Rp829dlwa`, you can use `admin:Rp829dlwa` to log on to the instance.

## Notes

- If you have enabled the VPC password-free access feature, no password is required when you access an ApsaraDB for Redis instance by using an internal endpoint. For more information, see VPC password-free access.
- If you have enabled the VPC password-free access feature, the password is still required when you access an ApsaraDB for Redis instance by using a public endpoint.
- For more information, see Troubleshooting for connection issues in ApsaraDB for Redis.

## Connect to an ApsaraDB for Redis instance

Run the following command in redis-cli to connect to an ApsaraDB for Redis instance.

```
redis-cli -h <host> -p <port> -a '<password>'
```

**Note** To use a direct connection to access an ApsaraDB for Redis instance, you must specify the `-c` parameter, as shown in the following example.

```
redis-cli -h <host> -p <port> -a '<password>' -c
```

## Parameters

| Parameter | Description |
| --- | --- |
| -h | Specifies the endpoint of the ApsaraDB for Redis instance.<br>• Private network connections: Access an ApsaraDB for Redis instance over a private network by using an internal endpoint. For more information, see View endpoints.<br>• Public network connections: Access an ApsaraDB for Redis instance over the Internet by using a public endpoint. For more information, see Use a public endpoint. |
| -p | Specifies the service port of an ApsaraDB for Redis instance. The default port number is 6379. You can change the port number. For more information, see Modify the port for the endpoint. |
| -a | The password that is used to connect to the ApsaraDB for Redis instance. Enclose the password with single quotation marks ('). For example, `'Rp829dlwa'`.<br>**Note** You can skip this parameter to avoid revealing the password in plaintext. After you run the preceding command, you can enter `auth <password>` for authentication, as shown in the following figure. |
| -c | This parameter is required when you use a direct connection to access an ApsaraDB for Redis cluster instance. |

Example


## FAQ

Why is the error message "ERR invalid password" returned when I use redis-cli to connect an instance?

The whitelist of an ApsaraDB for Redis instance is invalid. Make sure that you add a valid whitelist.

**Note** If the minor version of the instance is not the latest one, the returned message may be invalid. After you upgrade the instance to the latest minor version, the error message "ERR illegal address" is returned if the whitelist is not added as expected. For more information, see Upgrade the minor version.

# 4.4. Use a public endpoint

You can connect to an ApsaraDB for Redis instance by using a public endpoint. Compared with an internal endpoint, you may experience higher network latency if you use an public endpoint. We recommend that you connect to an ApsaraDB for Redis instance by using an internal endpoint in a production environment to ensure optimal service performance .

## Prerequisites

- The public IP address of the Elastic Compute Service (ECS) instance or an on-premises machine is added to the whitelist of the ApsaraDB for Redis instance. For more information, see Step 2: Set IP address whitelists.
- You cannot apply for public endpoints for instances of ApsaraDB for Redis 2.8 or 5.0 that have the password-free access feature enabled. We recommend that you disable this feature before you apply for public endpoints. For more information, see Enable password-free access.

  > **Note** You can apply for public endpoints for instances of ApsaraDB for Redis 4.0 that have the password-free access feature enabled. Then, you can use a private endpoint to access an ApsaraDB for Redis instance without a password. However, if you use a public endpoint to access an ApsaraDB for Redis instance, a password is still required.

- If you use a custom account to connect to the ApsaraDB for Redis instance, the format of the connection string must be `<user>:<password>`. For example, if the username of the custom account is `admin` and the password is `Rp829dlwa`, you can use `admin:Rp829dlwa` to log on to the ApsaraDB for Redis instance.

## Scenarios

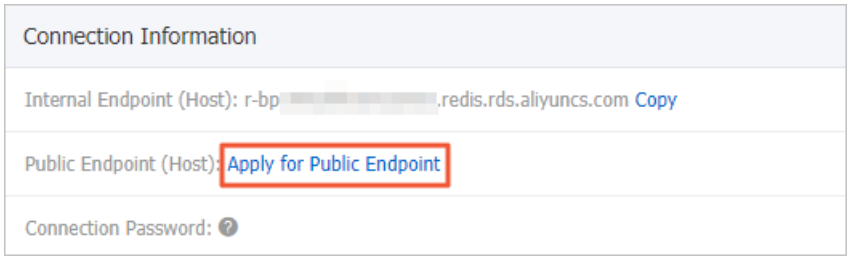
- Local access: You can access an ApsaraDB for Redis instance from an on-premises machine.
- Cross-account access: You can access ApsaraDB for Redis instances that are owned by other Alibaba Cloud accounts from your ECS instance.
- Cross-region access: You can create connections between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account but deployed in different regions.
- Cross-VPC access: You can create connections between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account and deployed in the same region, but in different virtual private clouds (VPCs).
- Cross-network access: You can create connections between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account and deployed in the same region, but of different network types.

## Pricing

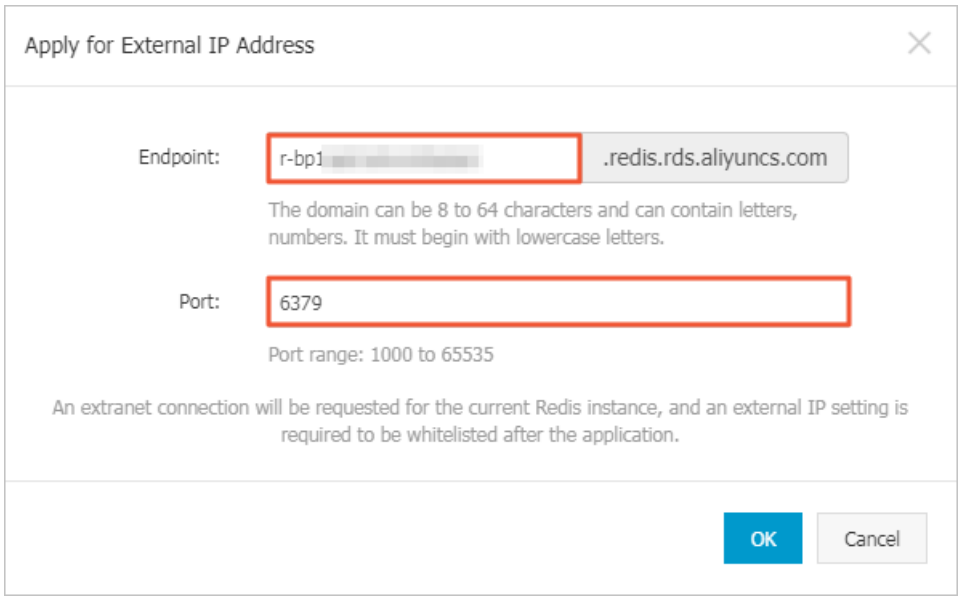
Public endpoints of ApsaraDB for Redis instances and Internet data transfer are free of charge.

## Apply for a public endpoint

1. Log on to the ApsaraDB for Redis console.
2. On the top of the page, select the region where the instance is deployed.
3. On the **Instances** page, click the Instance ID of the instance.
4. In the **Connection Information** section, click **Apply for Public Endpoint**.




5. In the dialog box that appears, enter an endpoint and a port number or use the default values.




6. Click **OK**.
   After the operation is complete, you can view the public endpoint in the **Connection Information** section.

> **Note** If you no longer use the public endpoint, click **Release Public Endpoint** to release the endpoint.

## Use a public endpoint to connect to an ApsaraDB for Redis instance.

You can use redis-cli or Redis clients of other languages to connect to ApsaraDB for Redis instances. For more information, see the following topics:

- Use redis-cli to connect to ApsaraDB for Redis
- Use a Redis client

## Solutions to connection issues over the Internet

- Make sure that you access an ApsaraDB for Redis instance through a public endpoint instead of an internal endpoint.
- You must add the public IP address of a client to the whitelist of the ApsaraDB for Redis instance. For more information, see Set IP address whitelists.
- For more information about how to fix issues of private network connections, see Troubleshooting for connection issues in ApsaraDB for Redis.

# 4.5. Use the sentinel compatibility mode to connect to ApsaraDB for Redis

ApsaraDB for Redis provides the sentinel compatibility mode. After you enable this mode, clients can connect to ApsaraDB for Redis instances in the same way as they connect to native Redis Sentinel.

## Prerequisites

- The ApsaraDB for Redis cluster instance is used.
- The engine version of the cluster instance is Redis 4.0 (Community Edition) or Redis 5.0 (Enhanced Edition).
- The sentinel compatibility mode is enabled. For more information, see Enable the Redis Sentinel-compatible mode.
- The internal IP address of the Elastic Compute Service (ECS) instance or the public IP address of the on-premises host is added to an IP address whitelist of the ApsaraDB for Redis instance. For more information, see Set IP address whitelists.

## Introduction to Redis Sentinel

Redis Sentinel provides Redis with services such as master and replica monitoring, fault alerting, and automatic failover. Redis Sentinel is used in many business scenarios that use on-premises Redis databases and require high reliability. To facilitate the migration of Redis databases to the cloud in such scenarios, Alibaba Cloud provides the Redis Sentinel-compatible mode.

> **Note** ApsaraDB for Redis uses the HA component developed by Alibaba Cloud, without the need to use Redis Sentinel.

After you enable the Sentinel-compatible mode, you can use the following commands:

| Statement | Description |
| --- | --- |
| SENTINEL sentinels | Queries Sentinel instances for a specified master and the status of these Sentinel instances. Follow this syntax:<br>`SENTINEL sentinels <the name of a master>` |
| SENTINEL get-master-addr-by-name | Queries the IP address and port number of a specified master. Follow this syntax:<br>`SENTINEL get-master-addr-by-name <the name of a master>` |

> **Note** Instances of ApsaraDB for Redis 2.8 do not support the preceding commands.

## Example: Create a password-free connection in sentinel mode

> **Note** To use the sentinel mode to connect to an ApsaraDB for Redis instance without a password, you must set the network type to VPC and enable password-free access. For more information, see Switch to VPC network and Enable password-free access.

In this example, spring-data-redis is used, as shown in the following source code:

```
@Bean
public JedisConnectionFactory connectionFactory() {
  RedisSentinelConfiguration sentinelConfig = new RedisSentinelConfiguration()
      .master("original-master-name")
      .sentinel(original-sentinel-1-host, original-sentinel-1-port)
      .sentinel(original-sentinel-2-host, original-sentinel-2-port);
  JedisPoolConfig poolConfig = new JedisPoolConfig();
  ...
  JedisConnectionFactory connectionFactory = new JedisConnectionFactory(sentinelConfig, poolConfig);
  return connectionFactory;
}
```

Parameters:

- master-name: the name of the master sentinel. You can specify a custom name.
- sentinel-host: the endpoint in the VPC network for an ApsaraDB for Redis instance.
- sentinel-port: the port number of an ApsaraDB for Redis instance. The default port number is 6379.

The following example shows how to configure the connection to use the sentinel compatibility mode to connect to ApsaraDB for Redis:

```
@Bean
public JedisConnectionFactory connectionFactory() {
  RedisSentinelConfiguration sentinelConfig = new RedisSentinelConfiguration()
      .master("any-name")
      .sentinel("r-********.redis.rds.aliyuncs.com", 6379);
  JedisPoolConfig poolConfig = new JedisPoolConfig();
  ...
  JedisConnectionFactory connectionFactory = new JedisConnectionFactory(sentinelConfig, poolConfig);
  return connectionFactory;
}
```

## Example: Create a connection that requires a password in sentinel mode

In this example, Jedis is used, as shown in the following source code:

```
String masterName = "original-master-name";
Set<String> sentinels = new HashSet<>();
sentinels.add("original-sentinel-1-host:original-sentinel-1-port");
sentinels.add("original-sentinel-2-host:original-sentinel-2-port");
GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
String dbPassword = "original-db-password";
String sentinelPassword = "original-sentinel-password";
JedisSentinelPool jedisSentinelPool =
    new JedisSentinelPool(masterName, sentinels, poolConfig,
        2000, 2000, dbPassword,
        0, null, 2000, 2000,
        sentinelPassword, null);
```

Parameters:

- masterName: the name of the master sentinel. You can specify a custom name.
- sentinels.add: Set the value to the endpoint and port number in the VPC network for an ApsaraDB for Redis instance. The format must be `r-********.redis.rds.aliyuncs.com:6379`.
- dbPassword/sentinelPassword: Set the dbPassword and sentinelPassword parameters to the password of the ApsaraDB for Redis instance. For more information about how to change your password, see Change the password.

> **Note** To use a custom account to connect to the ApsaraDB for Redis instance, the connection string must be in the `<user>:<password>` format. For example, the username of the custom account is admin and the password is Rp829dlwa, use `admin:Rp829dlwa` to log on to the instance.

The following example shows how to configure the connection to use the sentinel compatibility mode to connect to ApsaraDB for Redis:

```
String masterName = "any-name";
Set<String> sentinels = new HashSet<>();
sentinels.add("r-********.redis.rds.aliyuncs.com:6379");
GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
String dbPassword = "admin:Rp829dlwa";
String sentinelPassword = "admin:Rp829dlwa";
JedisSentinelPool jedisSentinelPool =
    new JedisSentinelPool(masterName, sentinels, poolConfig,
        2000, 2000, dbPassword,
        0, null, 2000, 2000,
        sentinelPassword, null);
```

# 5.Instructions

## 5.1. ApsaraDB for Redis console

The ApsaraDB for Redis console is a Web application used to manage ApsaraDB for Redis instances. You can perform basic operations for the instance management in the console. This topic describes how to use the console.

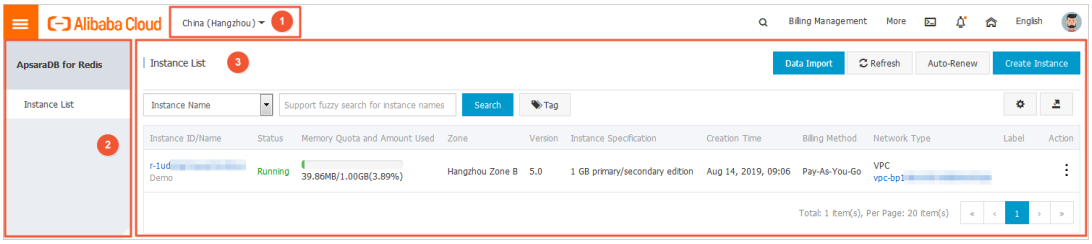
The ApsaraDB for Redis console is a part of the Alibaba Cloud console. For more information about general settings and basic operations in the console, see Alibaba Cloud console. This topic describes general settings in the ApsaraDB for Redis console. In case of any differences, follow the actual settings required in the console.

### Homepage

Log on to the ApsaraDB for Redis console to go to the homepage. The homepage displays the same information for ApsaraDB for Redis instances of all types.

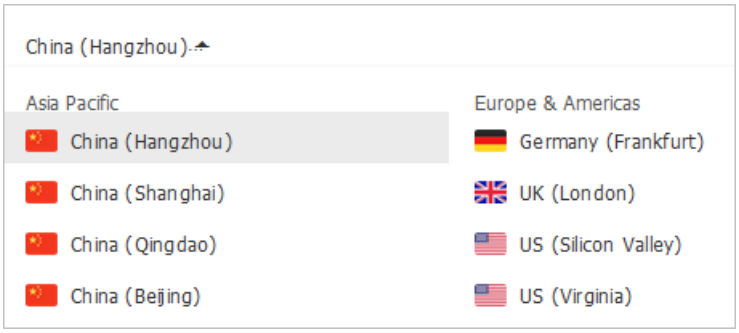
> **Note** If you do not have any Alibaba Cloud account, register an Alibaba Cloud account.

Homepage of the ApsaraDB for Redis console




The homepage provides the following features:

- Area 1 is the Region drop-down list. You can place your pointer on the drop-down list to show a list of regions. Click a target region to switch to the Instance List page for this region.




- Area 2 is the left-side navigation pane. When you log on to the ApsaraDB for Redis console, the **Instance List** page appears by default.
  - **Instance List**: shows a list of available resources in the current region. For more information, see the description of Area 3 in the following sections.
- Area 3 is **Instance List** page. The **Instance List** page displays the details of instances, such as **Instance ID**, **Status**, **Memory Quota and Amount Used**, **Zone**, **Version**, **Instance Specification**, **Creation Time**, **Billing Method**, and **Network Type**.

> **Note** **Memory Quota and Amount Used** shows the result of offline summary that the background system provides according to the collected information. A delay of approximately 10 minutes may exist between the summary and the current system status. Therefore, the result may be different from the actual value of the current time.
>
> To view real-time information, we recommend that you log on to Data Management Service (DMS). For more information, see Use DMS.

### Common features in the console

- Performance monitoring
- Alarm settings
- Whitelist settings
- Parameter settings
- Account management
- Backup and recovery
- Log management

# 5.2. Limits

This topic describes the limits on data types and some features of ApsaraDB for Redis.

| Item | Description |
| --- | --- |
| List data type | The number of lists is not limited. The maximum size of each element is 512 MB. We recommend that the number of elements in a list is less than 8,192 and the maximum length of the value does not exceed 1 MB. |
| Set data type | The number of sets is not limited. The maximum size of each element is 512 MB. We recommend that the number of elements in a set is less than 8,192 and the maximum length of the value does not exceed 1 MB. |
| Sorted set data type | The number of sorted sets is not limited. The maximum size of each element is 512 MB. We recommend that the number of elements in a sorted set is less than 8,192 and the maximum length of the value does not exceed 1 MB. |
| Hash data type | The number of fields is not limited. The maximum size of each element is 512 MB. We recommend that the number of elements in a hash table is less than 8,192 and the maximum length of the value does not exceed 1 MB. |

| Item | Description |
| --- | --- |
| Number of databases (DBs) | Each instance supports 256 databases.<br>**Note**<br>• The total size of data stored in all databases depends on the memory size of an instance.<br>• The system automatically assigns memory to a single DB based on the usage. The upper limit of assigned memory is the instance memory. For example, if DB 0 occupies all the memory, other databases have no data. |
| Supported Redis commands | For more information, see Overview. |
| Monitoring and alerts | ApsaraDB for Redis does not provide capacity alerts. You have to configure this feature in Cloud Monitor. For more information, see Alert settings.<br>We recommend that you set alerts for the following metrics: instance faults, instance failover, connection usage, failed operations, capacity usage, write bandwidth usage, and read bandwidth usage. |
| Policies to delete expired data | • Active expiration: the system periodically detects and deletes expired keys in the background.<br>• Passive expiration: the system deletes expired keys when you access these keys. |
| Idle connection recycling mechanism | ApsaraDB for Redis does not automatically recycle idle connections to ApsaraDB for Redis. You can manage the connections. |
| Data persistence policy | ApsaraDB for Redis uses the `AOF_FSYNC_EVERYSEC` policy and runs the fysnc command at a one-second interval. |

# 5.3. Overview

ApsaraDB for Redis provides instances of multiple editions, series, and architectures. The supported commands vary based on different instance types. This topic describes the commands supported and unsupported by ApsaraDB for Redis instances. You can refer to the topics in the following tables to view details of commands.

## Common commands

**Note** The common commands are applicable to the Community Edition and Enhanced Edition.

| Topic | Description |
| --- | --- |

| Topic | Description |
| --- | --- |
| Commands supported by Redis 2.8 | This topic describes the commands supported by ApsaraDB for Redis instances that use Redis 2.8. These instances include standard instances, cluster instances, and read/write splitting instances. ApsaraDB for Redis instances that use Redis 2.8 support the commands supported by native Redis 3.0. |
| Commands supported by Redis 4.0 | This topic describes the commands supported by ApsaraDB for Redis instances that use Redis 4.0. These instances include standard instances, cluster instances, and read/write splitting instances. |
| Unsupported commands | This topic describes the native Redis commands that are not supported by ApsaraDB for Redis. |
| Limits on the commands supported by cluster instances | This topic describes the limits on the commands supported by cluster instances. Cluster instances and standard instances are deployed in different architectures of ApsaraDB for Redis. These types of instances follow different rules to run Redis commands. |
| Redis commands developed by Alibaba Cloud | ApsaraDB for Redis also supports certain Redis commands developed by Alibaba Cloud. You can use these commands to manage cluster instances or read/write splitting instances of ApsaraDB for Redis. |

## Commands for Enhanced Edition

| Topic | Description |
| --- | --- |
| Commands supported by performance-enhanced instances | This topic describes the commands that are supported by ApsaraDB for Redis performance-enhanced instances. Performance-enhanced instances of ApsaraDB for Redis integrate with certain features of Tair, a distributed key-value storage system developed by Alibaba Group. Performance-enhanced instances support the commands supported by Community Edition as well as some new commands. This topic describes these new Redis commands. |
| CAS and CAD commands | This topic describes the enhanced commands that you can run to process strings on performance-enhanced instances of ApsaraDB for Redis Enhanced Edition (Tair). These commands include Compare And Set (CAS) and Compare And Delete (CAD). |
| TairString commands | This topic describes the commands supported by a TairString. |

| Topic | Description |
| --- | --- |
| TairHash commands | This topic describes the commands supported by a TairHash. |
| TairGIS commands | This topic describes the commands supported by a TairGIS. |
| TairBloom commands | This topic describes the commands supported by a TairBloom. |
| TairDoc commands | This topic describes the commands supported by a TairDoc. |
| Compatibility in commands | ApsaraDB for Redis hybrid-storage instances support most native Redis commands. To guarantee high performance, hybrid-storage instances have limits on the use of certain Redis commands. This topic describes these limited commands. |

# 5.4. Features of proxy servers

ApsaraDB for Redis instances of the cluster edition or read/write splitting edition use proxy servers to route commands, balance loads, and perform failover. This topic describes how proxy servers route commands, bring a read-only node offline, and handle special commands.

> **Note** The capabilities of the proxy servers in a cluster do not depend on only the absolute number of proxy servers due to the optimization and development of proxy servers. Alibaba Cloud ensures that the ratio of proxy servers meet the requirements described in the cluster specifications.

## Route commands in cluster instances

- Basic routing method
  - When receiving a command that operates a single key, proxy servers determine the hash slot to which the key belongs and route the command to the data shard where the hash slot resides.
  - When receiving a command that operates two or more keys stored on different shards, proxy servers split the command to multiple commands and route them to the corresponding shards. For more information about the commands that operate two or more keys, see Overview.
- Route Pub/Sub commands

  Proxy servers route Pub/Sub commands such as **PUBLISH** and **SUBSCRIBE** to the master node of the first data shard in the ApsaraDB for Redis cluster.

  > **Warning** Use Pub/Sub commands properly to avoid putting too much strain on the master node of the first data shard.
- Route commands with the idx parameter specified

When you run cluster commands developed by Alibaba Cloud, such as **IINFO**, **ISCAN**, **IMONITOR**, and **IMEMORY**, if you set the idx parameter to specify a shard, proxy servers route the commands to the specified data shard. For more information about these commands, see Overview.

## Route commands in read/write splitting instances

- Basic routing method
  - Proxy servers route write commands to the master node.
  - Proxy servers route read commands to the master node and read-only nodes based on their weights. By default, all nodes have the same weight. For example, if an instance has five read-only nodes, proxy servers route all read commands evenly to six nodes, including the master node and read-only nodes. Each node receives one-sixth of the commands.

    > Note **SLOWLOG** and **DBSIZE** are read commands.

- Route commands to the master node

  Proxy servers route the following commands to the master node:

  - Transaction commands: **MULTI** and **EXEC**
  - Lua scripting commands: **EVAL** and **EVALSHA**
  - **SCAN** and **INFO** commands
  - Pub/Sub commands, including **PUBLISH** and **SUBSCRIBE**

- Route commands to the first read-only node

  Proxy servers route the **HSCAN**, **SSCAN**, and **ZSCAN** commands to the first read-only node. If no read-only node is running, proxy servers route these commands to the master node.

- Route commands to a specified read-only node

  The **RIINFO**, **RIMONITOR**, and **RIMEMORY** commands are exclusive to ApsaraDB for Redis instances of the read/write splitting edition. When receiving these commands, proxy servers route them to the specified read-only node based on the idx and ro_slave_idx parameters. The idx parameter is used in cluster read/write splitting instances to specify a data shard. The ro_slave_idx parameter is used in both cluster and non-cluster read/write splitting instances to specify a read-only node. For more information about the commands that are exclusive for the read/write splitting instances, see Overview.

## Bring a read-only node offline

Proxy servers monitor the status of each read-only node in real time. Proxy servers can bring a read-only node offline to temporarily stop routing commands to the read-only node.

- When detecting that a read-only node is abnormal, proxy servers decrease the weight of the read-only node. If proxy servers fail to connect to the read-only node for a certain number of times, proxy servers bring the read-only node offline. Proxy servers bring the read-only node online again after the read-only node is resumed.
- When detecting that full data is being synchronized on a read-only node, proxy servers bring the read-only node offline until the synchronization is complete.

## Handle special commands

Typically, proxy servers create persistent connections to back-end data shards to process requests from users. If the requests contain the following special commands, proxy servers create additional connections to corresponding data shards to process subsequent requests.

- Blocking commands: **BRPOP**, **BRPOPLPUSH**, **BLPOP**, **BZPOPMAX**, and **BZPOPMIN**.
- Transaction commands: **MULTI**, **EXEC**, and **WATCH**.
- Monitoring commands: **MONITOR**, **IMONITOR**, and **RIMONITOR**.
- Sub commands: **SUBSCRIBE**, **UNSUBSCRIBE**, **PSUBSCRIBE**, and **PUNSUBSCRIBE**.

> **Note**
>
> - The maximum number of connections on each data shard is 10,000. Use the preceding commands properly in case that the number of connections exceeds the upper limit.
> - If you run the SCAN command to scan multiple databases, a large number of connections are created. Make sure that the number of connections does not exceed the upper limit.