

# Alibaba Cloud ApsaraDB for Redis

## Quick Start

Issue: 20190904

# 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 due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall 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 the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. 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 bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, 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 content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, 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 the Alibaba Cloud website, product programs, or content 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 contact Alibaba Cloud directly if you discover any errors in this document.



# Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 <b>Danger:</b> Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 <b>Warning:</b> Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 <b>Notice:</b> Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 <b>Note:</b> You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
<b>Bold</b>	It is used for buttons, menus, page names, and other UI elements.	Click <b>OK</b> .
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid <i>Instance_ID</i></code>
[ ] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand   slave}</code>



# Contents

---

Legal disclaimer.....	I
Generic conventions.....	I
1 Overview.....	1
2 Step 1: Create an instance.....	2
3 Step 2: Set IP whitelists.....	5
4 Step 3: Connect to the instance.....	9
4.1 Use a Redis client.....	9
4.2 Use redis-cli.....	21
4.3 Through the Internet.....	23
5 ApsaraDB for Redis console.....	26
6 Limits.....	28
7 Redis commands.....	30



# 1 Overview

---

## Purpose

This topic describes a series of operations from creating an ApsaraDB for Redis instance, connecting to databases of the instance, to managing the instance. In this way, you can easily understand the procedure of using the ApsaraDB for Redis instance.

## Intended audience

- Users that purchase an ApsaraDB for Redis instance for the first time.
- Users that want to know how to connect 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 [#unique\\_4](#) and [#unique\\_5](#).

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:

Figure 1-1: Flowchart for an ApsaraDB for Redis instance



## 2 Step 1: Create an instance

This topic describes how to create an ApsaraDB for Redis instance based on your business requirements.

### Procedure

1. Use one of the following methods to open the purchase page:
  - Open the ApsaraDB for Redis product page and click Buy Now.
  - Log on to the [ApsaraDB for Redis console](#) and click Create Instance in the upper-right corner.
2. Select a billing method.
  - **Subscription:** Pay for the service before using. You are charged when you create an instance. This billing method applies to long-term requirements. It is more cost-effective than the Pay-As-You-Go billing method. The longer the duration of the subscription you purchase, the higher the discounts.
  - **Pay-As-You-Go:** Pay for the service after using. You are charged by hour. This billing method applies to short-term requirements. You can release an instance when it is no longer used to save costs.





#### Notice:

You can switch the billing method of an instance from Pay-As-You-Go to subscription. However, you cannot switch the billing method from subscription to Pay-As-You-Go.

3. Set the following options.

Name	Description
Region	<p>Indicates the geo-location where the instance resides. You cannot change the region after you purchase the instance.</p> <ul style="list-style-type: none"> <li>· We recommend that you select a region in close proximity to the geographical location where you reside to guarantee the maximum access speed.</li> <li>· Make sure that an ApsaraDB for Redis instance resides in the same region as that of the ECS instance. Otherwise, both instances can only communicate with each other through the Internet rather than the Alibaba Cloud intranet, which may compromise the performance.</li> </ul>

Name	Description
Zone	Each zone is an independent geographical location that resides in a region. No difference exists between zones.
Network Type	<ul style="list-style-type: none"> <li>· <b>Classic Network:</b> traditional network.</li> <li>· <b>(Recommended) VPC:</b> indicates a type of new network provided by Alibaba Cloud. VPC is an abbreviation for Virtual Private Cloud. A VPC provides an isolated network environment with high security and performance over classic networks.</li> </ul> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;">  <b>Notice:</b> <ul style="list-style-type: none"> <li>· Ensure an identical network type for both the ApsaraDB for Redis instance and the ECS instance. Otherwise, the two instances cannot communicate with each other through the Alibaba Cloud intranet.</li> <li>· If you specify VPC as the network type for both the ApsaraDB for Redis instance and the ECS instance, make sure that both instances are in the same VPC. Otherwise, the two instances cannot communicate with each other through the Alibaba Cloud intranet.</li> <li>· You can change the network type of an ApsaraDB for Redis instance from classic network to VPC, see <a href="#">#unique_8</a>. However, you cannot change the network type of ApsaraDB for Redis instance from VPC to classic network.</li> </ul> </div>
VSwitch	A VSwitch is the basic network module for you to build a VPC. If no VSwitch is available in the VPC, we recommend that you <a href="#">#unique_9</a> .
Version	Supported versions for an ApsaraDB for Redis instance are listed as follows: <ul style="list-style-type: none"> <li>· 2.8</li> <li>· 4.0</li> <li>· 5.0</li> </ul>
Node Type	<ul style="list-style-type: none"> <li>· <b>Dual Copy:</b> provides a primary-secondary hot standby architecture for persistent data storage.</li> </ul>

Name	Description
Instance Class	<p>Each class corresponds to a set of configurations, such as the memory size, maximum number of connections, and bandwidth limit. For more information, see <a href="#">#unique_10</a>.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"> <p> <b>Note:</b>                      After you create an instance, a metadatabase is generated and occupies a low amount of storage space.</p> <ul style="list-style-type: none"> <li>· For a Standard version, the size of the metadatabase is about 32 MB.</li> <li>· For a Cluster version, the size of the metadatabase is calculated as follows: The number of shards included in a cluster × 32 MB.</li> </ul> </div>

4. Enter the Instance Name and select the Quantity. If you purchase a subscription instance, you need to select the Duration.



**Note:**

The connection password can be set after creating the instance, see [#unique\\_11](#).

5. On the right side of the page, click Buy Now.
6. On the Confirm Order page, confirm and select the ApsaraDB for KVStore (Pay-As-You-Go) Service Level Agreement (SLA).
7. Click Activate.



**Note:**

You will be prompted a message indicating a successful activation after you make the payment. The creation of a new ApsaraDB for Redis instance requires one to five minutes to complete. Then, you can find the new instance in the ApsaraDB for Redis console.

API operations

API	Description
<a href="#">#unique_12</a>	Creates an ApsaraDB for Redis instance.

## 3 Step 2: Set IP whitelists

---

To ensure database security and stability, before using an ApsaraDB for Redis instance, you must add one or more IP addresses or CIDR blocks that you use to connect to databases to a whitelist group of the instance. We recommend that you periodically check and adjust your whitelists to improve the access security protection and secure data in ApsaraDB for Redis.

### Prerequisites

The whitelist feature is applicable to certain kernel versions. If the kernel version of the instance does not support the whitelist feature, the system displays a prompt when you set a whitelist. In this case, you need to upgrade the minor version to the latest version. For more information, see [#unique\\_14](#).

### Procedure

1. Log on to the [ApsaraDB for Redis console](#).
2. On the menu bar, select the region where the target instance is located.
3. On the Instance List page, click the target instance ID, or click Manage in the Action column next to the target instance.
4. The Instance Information page is displayed by default. In the left-side navigation pane, click Whitelist Settings.
5. On the Whitelist Settings page, continue with one of these methods:
  - To customize the whitelist group name, create a new whitelist group:
    - a. Click Add a Whitelist Group in the upper-right corner.
    - b. In the Add a Whitelist Group dialog box that appears, set Group Name.



#### Note:

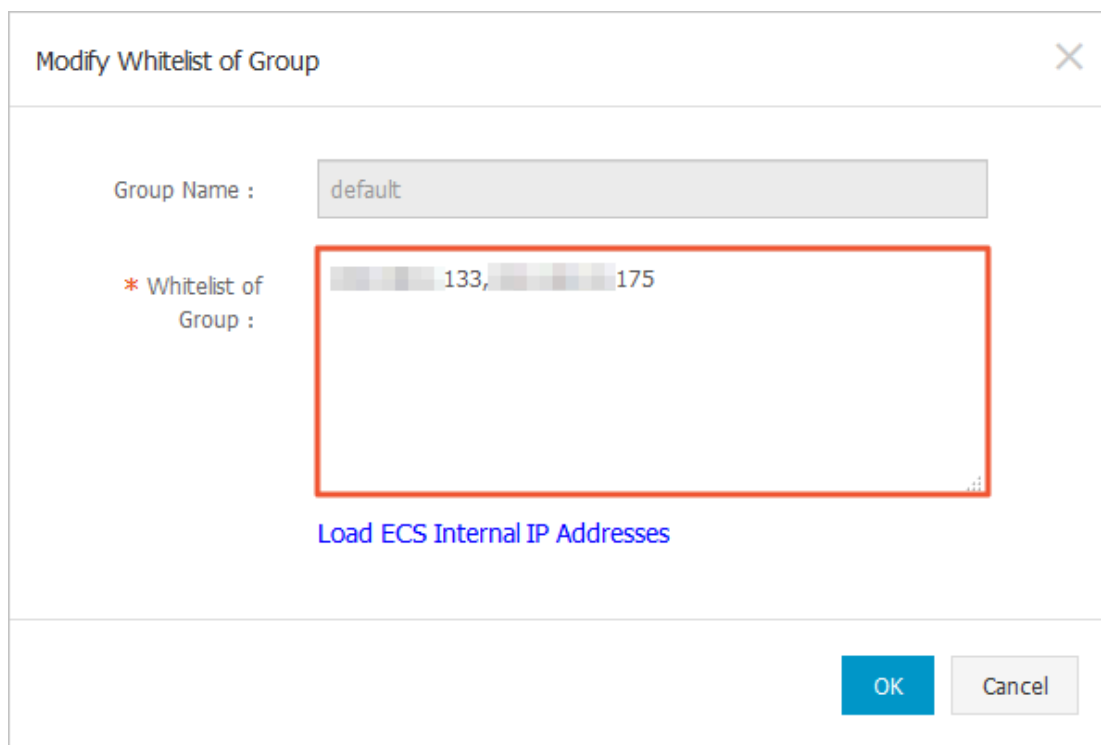
A group name must be 2 to 32 characters in length and contain lowercase letters, digits, or underscores (\_). The group name must start with a lowercase letter and end with a letter or digit. You cannot change this name after you create the whitelist group.

- If you do not require a custom whitelist group, click Modify next to the target whitelist group.

6. In the Add a Whitelist Group or Modify Whitelist of Group dialog box that appears, continue with one of these methods:

- Manually modify the Whitelist of Group field:
  - a. In the Whitelist of Group field, enter the IP addresses or CIDR blocks that you can use to connect to the ApsaraDB for Redis instance.

Figure 3-1: Manually modify the whitelist group



Modify Whitelist of Group

Group Name : default

\* Whitelist of Group : 133, 175

[Load ECS Internal IP Addresses](#)

OK Cancel



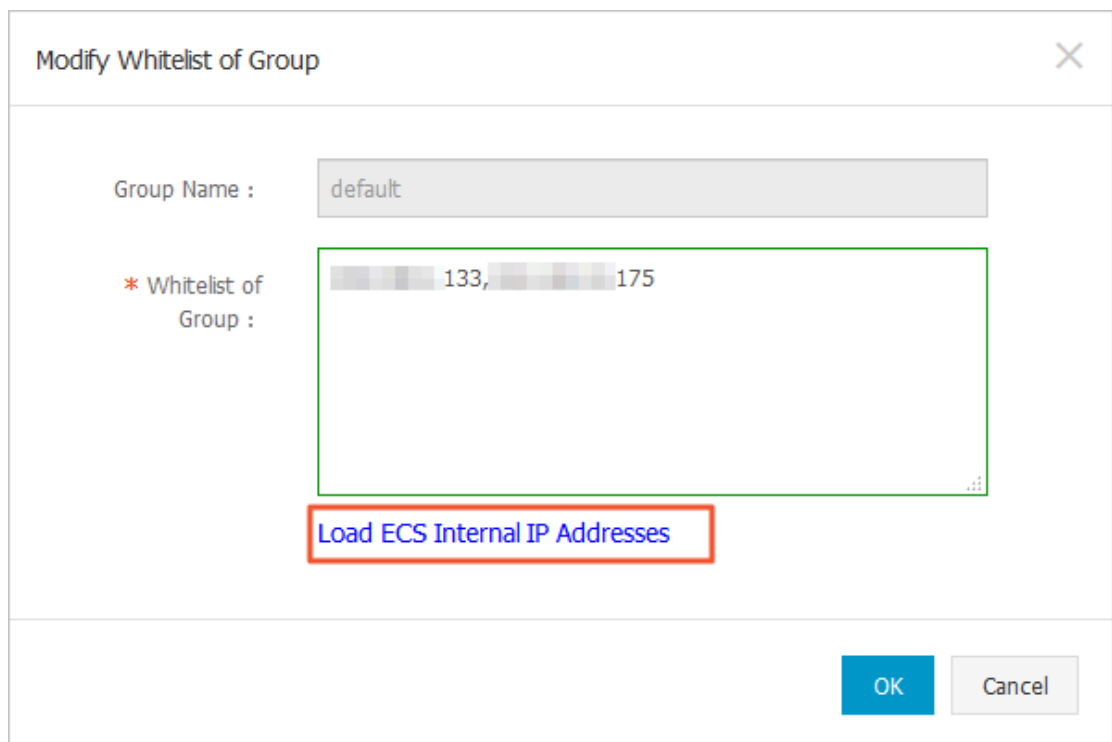
Note:

- Set the whitelist to `0 . 0 . 0 . 0 / 0` to allow connections from all IP addresses.
- Set the whitelist to `127 . 0 . 0 . 1` to block connections from all IP addresses.
- Set the whitelist to a CIDR block to allow connections from the IP addresses within the CIDR block, such as `10 . 10 . 10 . 0 / 24`.
- When you enter multiple IP addresses or CIDR blocks, separate them with commas (,) and leave no space before or after each comma.

- You can add 1,000 or fewer IP addresses or CIDR blocks to each whitelist group.

- Click OK.
- Load internal IP addresses of target ECS instances under the current Alibaba Cloud account:
    - Click Load ECS Internal IP Addresses.

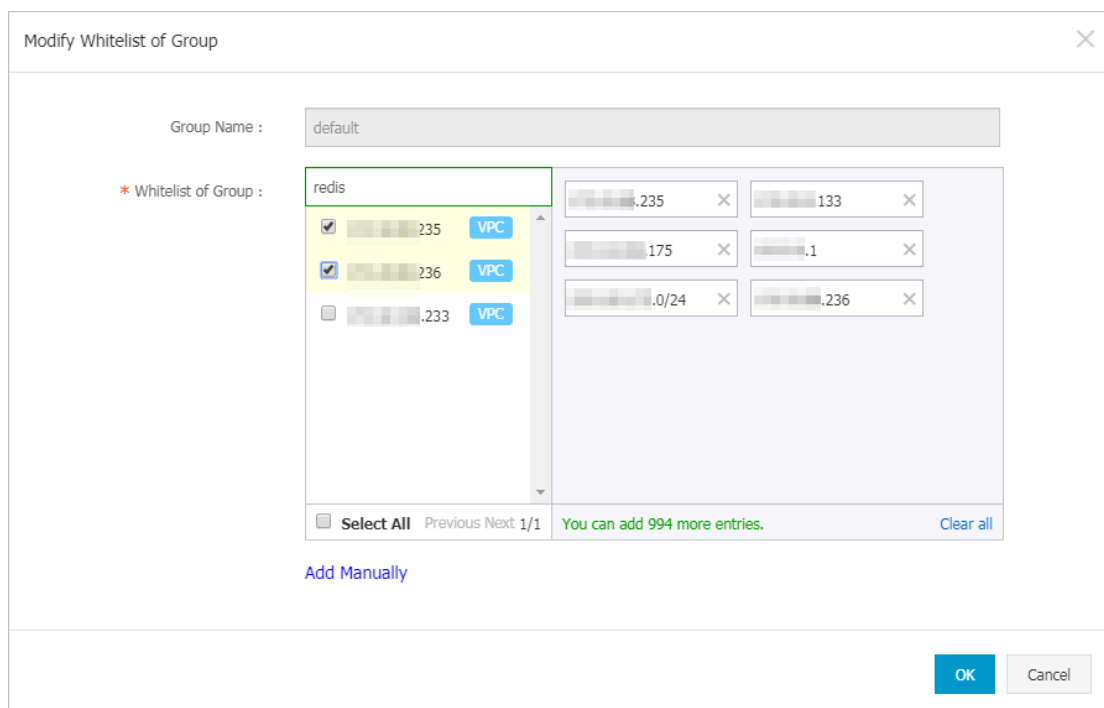
Figure 3-2: Load internal IP addresses of target ECS instances



The screenshot shows a dialog box titled "Modify Whitelist of Group" with a close button (X) in the top right corner. The "Group Name" field contains the text "default". Below it, the "Whitelist of Group" field is highlighted with a green border and contains the text "133, 175". A red rectangular box highlights the "Load ECS Internal IP Addresses" button, which is located below the whitelist field. At the bottom right of the dialog box, there are two buttons: "OK" (blue) and "Cancel" (gray).

**b. Select internal IP addresses of target ECS instances.**

**Figure 3-3: Select internal IP addresses of target ECS instances**



**Note:**

You can perform a fuzzy search by ECS instance name, ID, or IP address on the search bar above the list of ECS internal IP addresses.

**c. Click OK.**

**API operations**

Operation	Description
<a href="#">#unique_15</a>	Call this OpenAPI to query IP addresses or CIDR blocks that you can use to connect to a specified ApsaraDB for Redis instance.
<a href="#">#unique_16</a>	Call this OpenAPI to set IP whitelists of a specified ApsaraDB for Redis instance.



## 4 Step 3: Connect to the instance

---

### 4.1 Use a Redis client

You can connect to the ApsaraDB for Redis instance by using clients of several programming languages.

The database service of ApsaraDB for Redis is fully compatible with Redis database service. Therefore, you can connect to both database services in similar ways. All clients that are compatible with Redis protocols support connections to ApsaraDB for Redis. You can use any of these clients according to your application features.



#### Notice:

- If you enable password-free access for instances in the same VPC, you can connect to databases of the ApsaraDB for Redis instance with on password.
- Before you connect to the ApsaraDB for Redis instance by using a client, you must add the private IP address of the ECS instance to a [whitelist group of the ApsaraDB for Redis instance](#).

For more information about Redis clients, visit <http://redis.io/clients>.

- [Jedis client](#)
- [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 any of the following ways:

- Single Jedis connection
- JedisPool-based connection

To use a Jedis client to connect to an ApsaraDB for Redis instance, follow these steps:

1. Download and install the Jedis client. For more information, see [Jedis](#).

## 2. Example of single Jedis connection

a. Open 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 view the endpoint in the console .
int port = 6379 ;
Jedis jedis = new Jedis ( host , port ) ;
// Authentica tion informatio n
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 value .
String getvalue = jedis . get ( key ) ;
System . out . println ( " Get Key " + key + "
ReturnValu e : " + getvalue ) ;
jedis . quit ( ) ;
jedis . close ( ) ;
}
catch ( Exception e ) {
e . printStack Trace ( ) ;
}
}
}
```

b. Run the project. You have connected to ApsaraDB for Redis if you view the following result in the Eclipse console.

```
Set Key redis Value aliyun - redis
Get Key redis ReturnValu e aliyun - redis
```

Afterward, you can use your local 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

a. Open the Eclipse client, create a project, and then configure the pom file as follows:

```
< dependency >
< groupId > redis . clients </ groupId >
< artifactId > jedis </ artifactId >
< version > 2 . 7 . 2 </ version >
< type > jar </ type >
< scope > compile </ scope >
```

```
</ dependency >
```

**b. 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 ;
```

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

```
JedisPoolConfig config = new JedisPoolConfig ();
// Maximum number of idle connections . You can
// customize this parameter . Make sure that the
// specified maximum number of idle connections does
// not exceed the maximum number of connections that
// the ApsaraDB for Redis instance supports .
config . setMaxIdle ( 200 );
// Maximum number of connections . You can customize
// this parameter . Make sure that the specified
// maximum number of connections does not exceed the
// maximum number of connections that the ApsaraDB
// for Redis instance supports .
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 ();
```

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

```
JedisPoolConfig config = new JedisPoolConfig ();
// Maximum number of idle connections . You can
// customize this parameter . Make sure that the
// specified maximum number of idle connections does
```

```

not exceed the maximum number of connections that
the ApsaraDB for Redis instance supports .
config . setMaxIdle ( 200 );
// Maximum number of connections . You can customize
this parameter . Make sure that the specified
maximum number of connections does not exceed the
maximum number of connections that the ApsaraDB
for Redis instance supports .
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 );
    }
}
}

```

- e. Run the project. You have connected to ApsaraDB for Redis if you view the following result in the Eclipse console.

```

Set Key redis Value aliyun - redis
Get Key redis ReturnValu e aliyun - redis

```

Afterward, you can use your local Jedis client to manage your ApsaraDB for Redis instance.

### phpredis client

To use a phpredis client to connect to an ApsaraDB for Redis instance, follow these steps:

1. Download and install the phpredis client. For more information, see [phpredis](#).
2. In any editor that supports PHP editing, enter the following code:

```
<? php
```

```

/* Replace the following parameter values with the
host name and port number of the target instance .
*/
$ host = " localhost ";
$ port = 6379 ;
/* Replace the following parameter values with the
ID and password of the target instance . */
$ user = " test_usern ame ";
$ pwd = " test_passw ord ";
$ redis = new Redis ();
if ( $ redis -> connect ( $ host , $ port ) == false ) {
    die ( $ redis -> getLastErr or ());
}
if ( $ redis -> auth ( $ pwd ) == false ) {
    die ( $ redis -> getLastErr or ());
}
/* You can perform database operations after
authenticatio n . For more informatio n , visit https ://
github . com / phpre dis / phpre dis . */
if ( $ redis -> set ( " foo " , " bar " ) == false ) {
    die ( $ redis -> getLastErr or ());
}
$ value = $ redis -> get ( " foo " );
echo $ value ;
? >

```

3. Run the code. Afterward, you can use your local phpre dis client to connect to your ApsaraDB for Redis instance. For more information, visit <https://github.com/phpredis/phpredis>.

### redis-py client

To use a redis-py client to connect to an ApsaraDB for Redis instance, follow these steps:

1. Download and install the redis-py client. For more information, see [redis-py](#).
2. In any editor that supports Python editing, enter the following code. Afterward, you can use the local redis-py client to connect to the ApsaraDB for Redis instance and perform database operations.

```

#! / usr / bin / env python
#-*- coding : utf - 8 -*-
import redis
# Replace the following parameter values with the host
name and port number of the target instance .
host = ' localhost '
port = 6379
# Replace the following parameter value with the
password of the target instance .
pwd = ' test_passw ord '
r = redis . StrictRedi s ( host = host , port = port , password
= pwd )
# You can perform database operations after you
establish a connection . For more informatio n , visit
https :// github . com / andymccur d y / 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, follow these 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 );
/* Disconnect s 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. Test the code.

```

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, follow these 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 <https://github.com/ServiceStack/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 " ; /* IP address of
            the host that you want to connect to */
            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 that you want to connect to :
            Port */
            } ;
            RedisConfig . VerifyMasterConnections = false ; // You
            must set the parameter .
            PooledRedisClientManager redisPoolManager = new
            PooledRedisClientManager ( 10 /* Number of connections
            in the pool */ , 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 command. 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 " + key + " value " + getValue );
} 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, follow these 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:

In the code, the port field specifies the port of the ApsaraDB for Redis instance. Default value: 6379. The host field specifies the endpoint of the ApsaraDB for Redis instance. The following example shows the settings of the port and host fields:

```
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 instance returns
data of String type .
client . get ( " key " , function ( err , reply ) {
console . log ( reply . toString ( ) ); // print ` OK `
});
// If you specify a buffer , the instance returns 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, follow these steps:

1. Download and install [StackExchange.Redis](#).
2. Add a reference.

```
using StackExchange . Redis ;
```

### 3. Initialize ConnectionMultiplexer.

ConnectionMultiplexer is the core of StackExchange.Redis, and shared 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 : 6379 ,
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 .
IsConnecte d )
            {
                redisConn = Connection Multiplexe r . Connect
( configurat ionOptions );
            }
        }
        return redisConn ;
    }
}

```

**Note:**

**ConfigurationOptions** contains multiple options, such as **keepAlive**, **connectRetry**, and **name**. For more information, see [StackExchange.Redis.ConfigurationOptions](#).

4. **GetDatabase()** returns a lightweight object. You can obtain this object from the object of **ConnectionMultiplexer**.

```

redisConn = getRedisCo nn ();
var db = redisConn . GetDatabas e ();

```

5. The following examples show five types of data structures. The API operations used in these examples are different from their usage in the native Redis service. These data structures include: **string**, **hash**, **list**, **set**, and **sortedset**.

- **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 counterVal ue = db . StringIncr ement ( counterKey );
Console . WriteLine ( " incr " + counterKey + " , result is
" + counterVal ue );
// 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 " );
HashSet [] values = db . HashGetAll ( hashKey );
// hgetall
Console . Write ( " hgetall " + hashKey + " , result is " );
for ( int i = 0 ; i < values . Length ; i ++ )
{
    HashSetEntry 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 );
```

- **sortedset**

```
string  sortedSetKey = " myZset ";
// sadd
db . SortedSetAdd ( sortedSetKey , " xiaoming " , 85 );
db . SortedSetAdd ( sortedSetKey , " xiaohong " , 100 );
db . SortedSetAdd ( sortedSetKey , " xiaofei " , 62 );
db . SortedSetAdd ( sortedSetKey , " xiaotang " , 73 );
// zrevrangeb yscore
```

```
RedisValue [] names = db.SortedSetRangeByRank(
    sortedSetKey, 0, 2, Order.Ascending);
Console.WriteLine("zrevrangebyscore " + sortedSetKey + "
    0 2, result is ");
for (int i = 0; i < names.Length; i++)
{
    Console.WriteLine(names[i] + " ");
}
Console.WriteLine();
```

## 4.2 Use redis-cli

You can use the Redis command-line interface (`redis-cli`) to connect to ApsaraDB for Redis.

### Introduction to redis-cli

`Redis-cli` is a command-line tool included in the Redis software distribution. You can use `redis-cli` to connect to an ApsaraDB for Redis instance and send commands to the instance to manage data.

With `redis-cli`, you can connect to an ApsaraDB for Redis instance from an ECS instance running Linux through the Alibaba Cloud intranet, or from a local host through the Internet. Accessing ApsaraDB for Redis through the Alibaba Cloud intranet provides higher security and performance.

To access an ApsaraDB for Redis instance from a local host through the public network, you must follow the instructions in [#unique\\_21](#) to apply for a public endpoint. Then, you can follow the instructions in the [How to connect](#) section of this topic to connect to the instance.

### Install redis-cli

Install the Redis software distribution that includes `redis-cli` in Linux. For more information, see [Redis community](#).

### Prerequisites

#### Connect through the Alibaba Cloud intranet

- If the network type for both the ECS instance and the ApsaraDB for Redis instance is VPC, the two instances must reside in the same VPC of a region.
- If the network type for both the ECS instance and the ApsaraDB for Redis instance is classic network, the two instances must reside in the same region.
- You have added the private IP address of an ECS instance to the whitelist of an ApsaraDB for Redis instance.

- You have installed the Redis software distribution on the ECS instance.

### Connect through the Internet

- The ApsaraDB for Redis instance has a public endpoint. For more information, see [#unique\\_21](#).
- You have added the public IP address of the local host to a [whitelist](#) of the ApsaraDB for Redis instance.
- The operating system of the local host must be Linux.
- You have installed the Redis software distribution on the local host.

### Precautions

- If you access an ApsaraDB for Redis instance from its private endpoint with the [VPC password-free access](#) feature enabled, you can access the instance without a password.
- If you access an ApsaraDB for Redis instance from its public endpoint with the [VPC password-free access](#) feature enabled, a password is still required for accessing the instance.

### How to connect

You can use the following command to connect to an ApsaraDB for Redis instance.

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

Table 4-1: Parameters

Name	Description
-h	<p>Specifies the endpoint of the ApsaraDB for Redis instance.</p> <ul style="list-style-type: none"> <li>• Access an ApsaraDB for Redis instance through the Alibaba Cloud intranet, use an <a href="#">private endpoint</a>.</li> <li>• Access an ApsaraDB for Redis instance through the Internet, use a <a href="#">public endpoint</a>.</li> </ul>
-p	<p>Specifies the service port of the ApsaraDB for Redis instance.</p> <ul style="list-style-type: none"> <li>• The default port number is 6379. You cannot change the port number.</li> </ul>

Name	Description
-a	Set the password for the ApsaraDB for Redis instance. You can skip this parameter to avoid displaying a password in plain text to enhance security. After running the preceding command, you can enter <code>auth &lt; password &gt;</code> to complete the authentication. The following figure shows an example.

Figure 4-1: Command example

```
[root@ ~]# redis-cli -h r-bpl...redis.rds.aliyuncs.com -p 6379
r-bpl...redis.rds.aliyuncs.com:6379> auth a
OK
r-bpl...redis.rds.aliyuncs.com:6379>
```

## 4.3 Through the Internet

Public endpoints for ApsaraDB for Redis instances are also known as Internet endpoints. You can access an Apsara for Redis instance through the Internet with its public endpoint. However, you may experience high network latency if you use a public endpoint. In production environment, we recommend that you connect to an ApsaraDB for Redis instance through a private endpoint to guarantee high performance of the service.

### Prerequisites

- The public IP address of an ECS instance or a local host has been added to the whitelist of the ApsaraDB for Redis instance. For more information about how to configure the whitelist, see [#unique\\_22](#).
- For ApsaraDB for Redis 2.8 or 5.0 instances, you cannot apply for public endpoints with the [password-free access](#) feature enabled. Please disable password-free access before applying for public endpoints.



#### Note:

For ApsaraDB for Redis 4.0 instances, you can apply for public endpoints after enabling the password-free access feature. At this point, you can access an ApsaraDB for Redis instance from a private endpoint without a password. However, a password is still required to access an ApsaraDB for Redis instance from a public endpoint.

### Scenarios

- **Local access:** You can access an ApsaraDB for Redis instance from a local host.

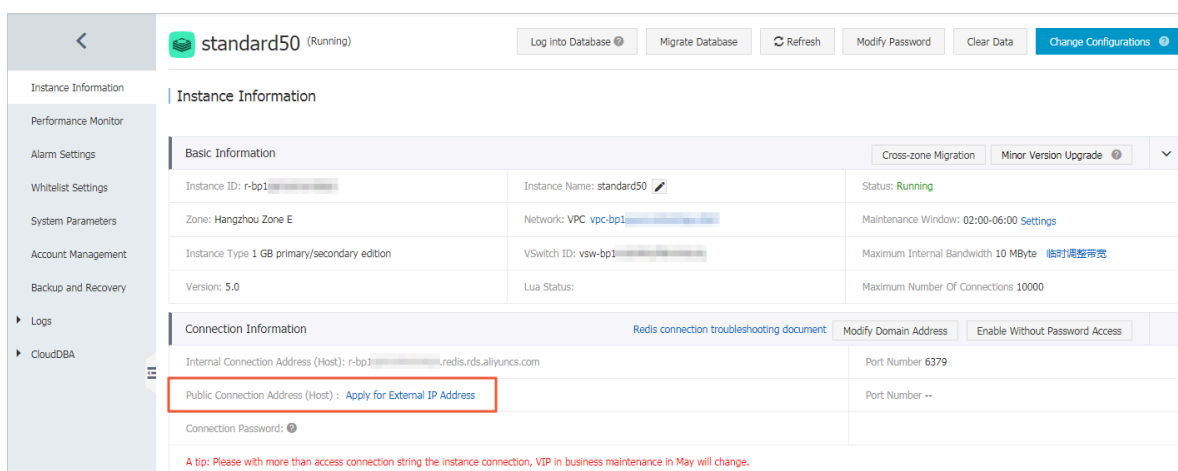
- **Cross-account access:** You can access ApsaraDB for Redis instances owned by other Alibaba Cloud accounts from your ECS instance.
- **Cross-region access:** You can have reciprocal access between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account, but reside in different regions.
- **Cross-VPC access:** You can have reciprocal access between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account and reside in the same region, but in different VPCs.
- **Cross-network access:** You can have reciprocal access between an ECS instance and an ApsaraDB for Redis instance. The two instances are owned by the same Alibaba Cloud account and reside in the same region but have different network types.

## Pricing

Public endpoints for ApsaraDB for Redis instances and generated public traffic are free of charge.

## Apply for a public endpoint

1. Log on to the [ApsaraDB for Redis console](#).
2. In the upper-left corner, on the right side of the Alibaba Cloud trademark, select the region where the target instance resides.
3. On the Instance List page, click the target instance ID or click Manage in the Actions column corresponding to the target instance.
4. On the Instance Information page, click Apply for External IP Address in the Connection Information area.



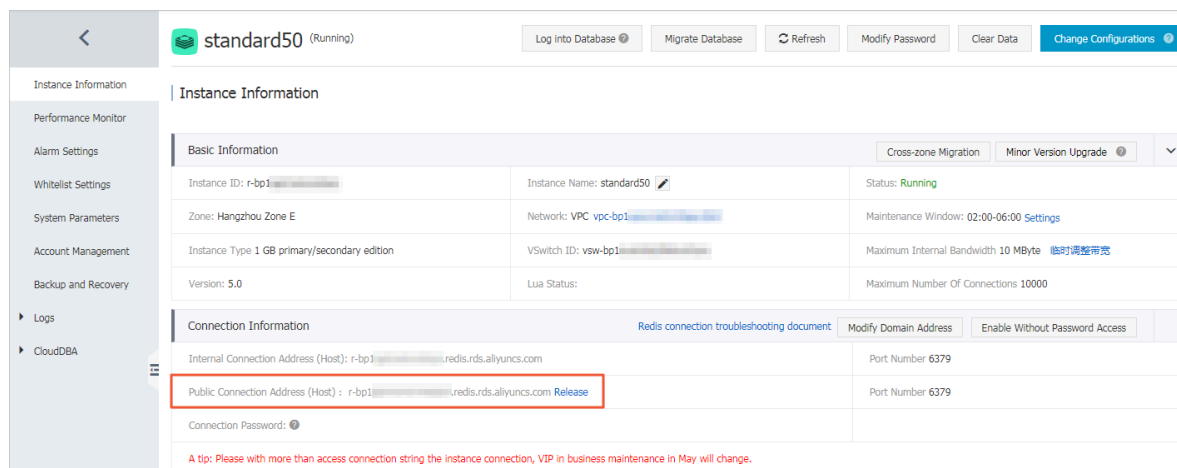
The screenshot displays the 'Instance Information' page for a Redis instance named 'standard50'. The instance is in a 'Running' state. The 'Basic Information' section includes details such as Instance ID, Instance Name, Zone (Hangzhou Zone E), Network (VPC), Instance Type (1 GB primary/secondary edition), Version (5.0), and Lua Status. The 'Connection Information' section is expanded, showing the Internal Connection Address (Host) and Port Number (6379). A red box highlights the 'Public Connection Address (Host)' field, which has a button labeled 'Apply for External IP Address' next to it. A red tip at the bottom of the page reads: 'A tip: Please with more than access connection string the instance connection, VIP in business maintenance in May will change.'

5. In the Apply for External IP Address dialog box, enter the endpoint and port number, and click OK.



6. On the Instance Information page, view the Public Endpoint in the Connection Information area.

Figure 4-2: Public endpoints for ApsaraDB for Redis instances



**Note:**  
 If a public endpoint is no longer used, click **Release Public Endpoint** next to the **Public Endpoint** to release the endpoint.

Connect to an instance by using a public endpoint

You can use DMS, redis-cli, or Redis clients in various languages to connect to the Redis instance. For more information about the connection methods, see the following topics:

- [#unique\\_26](#)
- [#unique\\_27](#)

Resolve connection issues through the public network

- Make sure the endpoint you use is the public endpoint rather than the private endpoint. Check [this picture](#) to see where the public endpoint is.
- You must add the public IP address of a client to the [whitelist of the ApsaraDB for Redis instance](#).

## 5 ApsaraDB for Redis console

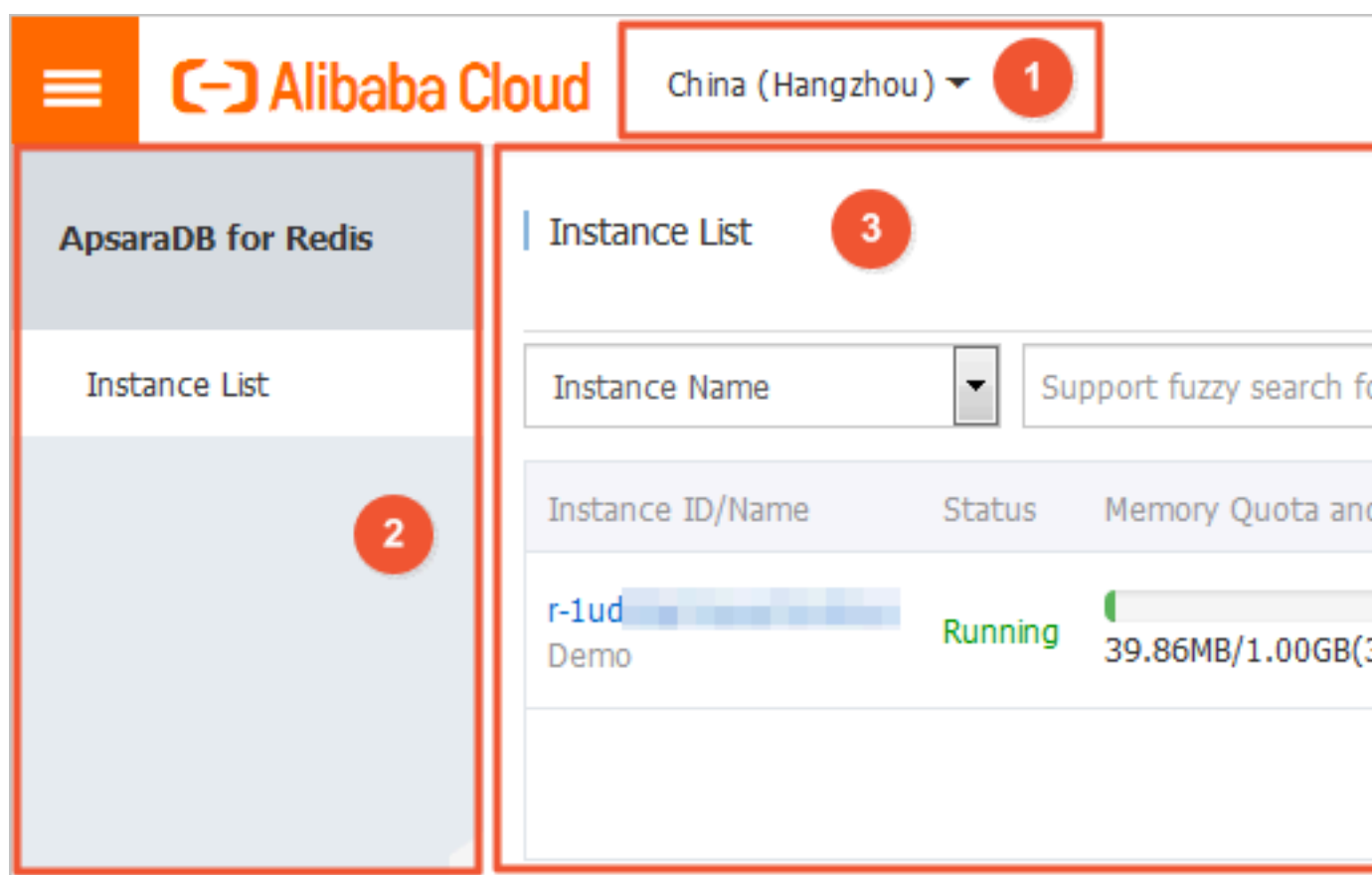
The ApsaraDB for Redis console is a Web application used to manage ApsaraDB for Redis instances. You can perform basic operations for the instance management in the console. This topic describes how to use the console.

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.

Figure 5-1: 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.

#### Common features in the console

- [Performance monitoring](#)
- [Alarm settings](#)
- [Whitelist settings](#)
- [Parameter settings](#)
- [Backup and recovery](#)

## 6 Limits

Item	Description
List data type	The number of lists is not limited. The size of each element is 512 MB or less. We recommend that the number of elements in a list is less than 8,192. The value length is 1 MB or less.
Set data type	The number of sets is not limited. The size of each element is 512 MB or less. We recommend that the number of elements in a set is less than 8,192. The value length is 1 MB or less.
Sorted set data type	The number of sorted sets is not limited. The size of each element is 512 MB or less. We recommend that the number of elements in a sorted set is less than 8,192. The value length is 1 MB or less.
Hash data type	The number of fields is not limited. The size of each element in a hash table is 512 MB or less. We recommend that the number of elements in a hash table is less than 8,192. The value length is 1 MB or less.
Number of databases (DBs)	<p>Each instance supports 256 databases.</p> <div style="background-color: #f0f0f0; padding: 10px;">  <b>Note:</b> <ul style="list-style-type: none"> <li>· The total size of data stored in all databases depends on the memory size of an instance.</li> <li>· 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.</li> </ul> </div>
Supported Redis commands	For more information, see <a href="#">#unique_6</a> .

Item	Description
Monitoring and alerts	<p>ApsaraDB for Redis does not provide capacity alerts. You have to configure this feature in CloudMonitor. For more information about the configuration, see <a href="#">ApsaraDB for Redis</a>.</p> <p>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.</p>
Expired data deletion policies	<ul style="list-style-type: none"> <li>· Active expiration: the system periodically detects and deletes expired keys in the background.</li> <li>· Passive expiration: the system deletes expired keys when you access these keys.</li> </ul>
Idle connection recycling mechanism	<p>ApsaraDB for Redis does not actively recycle idle connections to ApsaraDB for Redis. You can manage the connections.</p>
Data persistence policy	<p>ApsaraDB for Redis uses the <code>AOF_FSYNC_</code> <code>EVERYSEC</code> policy, and runs the <code>fsync</code> command at a one-second interval.</p>

## 7 Redis commands

This topic describes Redis commands that engine versions 2.8 and 4.0 support, and unsupported and restricted Redis commands. ApsaraDB for Redis is compatible with Redis 3.0 and supports Redis 3.0 GEO commands.

### Supported Redis commands

Table 7-1: Supported Redis commands

Keys	String	Hash	List	Set	SortedSet
DEL	APPEND	HDEL	BLPOP	SADD	ZADD
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD
EXISTS	BITOP	HGET	BRPOPLPUSH	SDIFF	ZCOUNT
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY
EXPIREAT	DECR	HINCRBY	LINSERT	SINTER	ZRANGE
MOVE	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE
PERSIST	GET	HKEYS	LPOP	SISMEMBER	ZRANK
PEXPIRE	GETBIT	HLEN	LPUSH	SMEMBERS	ZREM
PEXPTREAT	GETRANGE	HMGET	LPUSHX	SMOVE	ZREMRANGEBYRANK
PTTL	GETSET	HMSET	LRANGE	SPOP	ZREMRANGEBYSCORE
RANDOMKEY	INCR	HSET	LREM	SRANDMEMBER	ZREVRANGE
RENAME	INCRBY	HSETNX	LSET	SREM	ZREVRANGEBYSCORE
RENAMENX	INCRBYFLOAT	HVALS	LTRIM	SUNION	ZREVRANK
RESTORE	MGET	HSCAN	RPOP	SUNIONSTORE	ZSCORE
SORT	MSET		RPOPLPUSH	SSCAN	ZUNIONSTORE
TTL	MSETNX		RPUSH		ZINTERSTORE

Keys	String	Hash	List	Set	SortedSet
TYPE	PSETEX		RPUSHX		ZSCAN
SCAN	SET				ZRANGEBYLEX
OBJECT	SETBIT				ZLEXCOUNT
	SETEX				ZREMRANGEBYLEX
	SETNX				
	SETRANGE				
	STRLEN				

Table 7-2: Supported Redis commands

HyperLogLog	Pub/Sub	Transaction	Connection	Server	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	FLUSHALL	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	FLUSHDB	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	DBSIZE	SCRIPT EXISTS	GEOPOS
	PUNSUBSCRIBE	UNWATCH	QUIT	TIME	SCRIPT FLUSH	GEODIST
	SUBSCRIBE	WATCH	SELECT	INFO	SCRIPT KILL	GEORADIUS
	UNSUBSCRIBE			KEYS	SCRIPT LOAD	GEORADIUSBYMEMBER
				CLIENT KILL		
				CLIENT LIST		
				CLIENT GETNAME		
				CLIENT SETNAME		
				CONFIG GET		

HyperLogLog	Pub/Sub	Transaction	Connection	Server	Scripting	Geo
				MONITOR		
				SLOWLOG		



**Note:**

- When you run the CLIENT LIST command on a Redis cluster instance, you can retrieve a list of all client connections to the specified proxy. In this list, the id, age, idle, addr, fd, name, db, multi, omem, and cmd fields are described in the same way as those in the Redis kernel. The sub and psub fields are not distinguished for the proxy, and their values are either 1 or 0 at the same time. The qbuf, qbuf-free, obl, and oll fields are not described.
- You can run the CLIENT KILL command on a Redis cluster instance in two ways: `client kill ip : port` and `client kill addr ip : port`.

**New Redis commands for engine version 4.0**

Table 7-3: New Redis commands for engine version 4.0

Keys	Server
UNLINK	SWAPDB
	MEMORY

**Updated Redis commands for engine version 4.0**

The FLUSHALL and FLUSHDB commands support the ASYNC option.



**Note:**

Based on this option, you can run the FLUSHALL or FLUSHDB command asynchronously in a new thread. In this way, this thread cannot block the service. For more information about features of engine version 4.0, see [Features of engine version 4.0 of ApsaraDB for Redis](#).

**Unsupported Redis commands**

Keys	Server
MIGRATE	BGREWRITEAOF



Keys	Server
	BGSAVE
	CONFIG REWRITE
	CONFIG SET
	CONFIG RESETSTAT
	COMMAND
	COMMAND COUNT
	COMMAND GETKEYS
	COMMAND INFO
	DEBUG OBJECT
	DEBUG SEGFAULT
	LASTSAVE
	ROLE
	SAVE
	SHUTDOWN
	SLAVEOF
	SYNC

#### Redis commands restricted by cluster instances

Keys	Strings	Lists	HyperLogLog	Transaction	Scripting
RENAME	MSETNX	RPOPLPUSH	PFMERGE	DISCARD	EVAL
RENAMENX		BRPOP	PFCOUNT	EXEC	EVALSHA
SORT		BLPOP		MULTI	SCRIPT EXISTS
		BRPOPLPUSH		UNWATCH	SCRIPT FLUSH
				WATCH	SCRIPT KILL
					SCRIPT LOAD



**Note:**

- Restricted commands only support scenarios where target keys are distributed in a single hash slot. You cannot merge data from multiple hash slots. Therefore, you need to use hash tags to distribute all target keys to only one hash slot.

For example, when you process three keys, key1, aakey, and abkey3, you need to store them as {key}1, aa{key}, and ab{key}3 to effectively call restricted commands. For more information about how to use hash tags, visit <http://redis.io/topics/cluster-spec>.

- If you do not run the WATCH command prior to a transaction, and each command in the transaction processes only one key, these keys can be in different slots. You can run these commands in the same way as you run them in a directly connected database. In other scenarios, all keys that all commands process in a transaction must be in the same slot.
  - The commands that process multiple keys include: DEL, SORT, MGET, MSET, BITOP, EXISTS, MSETNX, RENAME, RENAMENX, BLPOP, BRPOP, RPOPLPUSH, BRPOPLPUSH, SMOVE, SUNION, SINTER, SDIFF, SUNIONSTORE, SINTERSTORE, SDIFFSTORE, ZUNIONSTORE, ZINTERSTORE, PFMERGE, and PFCOUNT.
  - The commands that do not support transactions include: WATCH, UNWATCH, RANDOMKEY, KEYS, SUBSCRIBE, UNSUBSCRIBE, PSUBSCRIBE, PUNSUBSCRIBE, PUBLISH, PUBSUB, SCRIPT, EVAL, EVALSHA, SCAN, ISCAN, DBSIZE, ADMINAUTH, AUTH, PING, ECHO, FLUSHDB, FLUSHALL, MONITOR, IMONITOR, RIMONITOR, INFO, IINFO, RIINFO, CONFIG, SLOWLOG, TIME, and CLIENT.

### Lua restrictions

You can directly use Lua scripts for the standard master-replica edition and the standard single-node edition.

Lua scripts support the cluster edition in the following conditions:

- You must use KEYS arrays to pass all keys. For Redis commands in redis.call() and redis.pcall(), keys must be KEYS arrays. You cannot replace KEYS with Lua variables. Otherwise, the system returns the following error: `- ERR bad lua script for redis cluster , all the keys that the script uses should be passed using the KEYS array \r\n`.

- All keys must be in the same slot. Otherwise, the system returns the following error: `"- ERR eval / evalsha command keys must be in same slot \ r \ n "`.
- You must use keys when running Redis commands for cluster instances. Otherwise, the system returns the following error: `"- ERR for redis cluster , eval / evalsha number of keys can ' t be negative or zero \ r \ n "`.

#### Proprietary Redis commands for cluster instances

- **INFO KEY:** you can run this command to query slots and databases (DBs) that keys belong to. The native Redis command INFO can contain only one optional section in this way: `info [ section ]`. When you run some commands for cluster instances of ApsaraDB for Redis, all keys must be in the same slot. The `INFO KEY` command allows you to check whether keys are in the same slot or DB. You can run this command in this way:

```
127 . 0 . 0 . 1 : 6379 > info key test_key
slot : 15118 node_index : 0
```



#### Notice:

- In earlier editions, the `INFO KEY` command may return a different `node index` from the `node index` in the topology of an instance. This issue has been fixed in the latest edition.
  - The `INFO KEY` command returns shard servers of cluster instances. These shard servers are different from the DBs used in the `SELECT` command.
- **IINFO:** you can run this command to specify the node of ApsaraDB for Redis to run the INFO command. This command is similar to the INFO command. You can run this command in this way:

```
iinfo db_idx [ section ]
```

In this command, `db_idx` supports the range of `[0, nodecount]`. You can obtain the `nodecount` value by running the `INFO` command, and specify the `section` option in the same way as you specify this option for a Redis database. For more information about a node of ApsaraDB for Redis, you can run the `IINFO` command or check the instance topology in the console.

- **RIINFO:** you can run this command in a similar way as you run IINFO, but only in read/write splitting scenarios. This command specifies the `idx` value as the identifier of the read-only replica node where you want to run the INFO command. If you use this command on instances other than read/write splitting cluster instances, the system returns an error. You can run this command in this way:

```
riinfo db_idx ro_slave_idx [ section ]
```

- **ISCAN:** you can run this command to specify the DB of a cluster where you want to run the SCAN command. This command provides the `db_idx` parameter on the basis of SCAN. The `db_idx` parameter supports the range of [0, nodecount]. You can obtain the nodecount value by running the INFO command or by checking the instance topology in the console. You can run this command in this way:

```
iscan db_idx cursor [ MATCH pattern ] [ COUNT count ]
```

- **IMONITOR:** similar to IINFO and ISCAN, this parameter provides the `db_idx` parameter on the basis of the MONITOR command. The `db_idx` parameter specifies the node where you want to run MONITOR. The `db_idx` parameter supports the range of [0, nodecount]. You can obtain the nodecount value by running the INFO command or by checking the instance topology in the console. You can run this command in this way:

```
imonitor db_idx
```

- **RIMONITOR:** similar to RIINFO, you can run this command to specify the read-only replica node in a specified shard where you want to run the MONITOR command. This command supports read/write splitting scenarios. You can run this command in this way:

```
rmonitor db_idx ro_slave_idx
```



#### Note:

Before you run IMONITOR and RIMONITOR, use telnet to verify connection conditions. To exit these commands, run the `QUIT` command.

#### Notes

- For more information about Redis commands, see [Redis documents](#).
- If the system returns the `unknown command` error when you run a supported command on a cluster instance, you need to [upgrade the minor version](#) in the console.