阿里云
云数据库 Redis 版

快速入门

文档版本：20190902

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 快速入门概览

文档目的

快速入门旨在介绍如何创建Redis实例以及连接实例数据库，使用户能够了解从创建Redis实例到连接并管理实例的流程。

目标读者

- 首次购买Redis实例的用户
- 想要了解如何连接Redis实例的用户

快速入门流程图

若您初次使用云数据库Redis版，请先了解#unique_4以及#unique_5。

云数据库Redis版完全兼容原生Redis命令，并新增了部分自研命令以提供更优质的服务。各版本命令的支持情况与自研命令的说明请参见Redis命令。

通常，从新购实例到可以开始使用实例，您需要完成如下操作：

图 1-1: Redis快速入门流程图

2 步骤1：创建实例

您可以根据本文的步骤创建适应业务需求的云数据库Redis版实例。

操作步骤

1. 使用下列方法中任意一种打开购买页：
   - 打开云数据库Redis版产品首页，单击立即购买。

     说明：
     如果尚未登录阿里云账号，单击立即购买后需要先使用阿里云账号和密码登录。
   - 登录Redis管理控制台，单击右上角的创建实例。
2. 选择计费方式。
   - 包年包月：属于预付费，即在新建实例时需要支付费用。适合长期需求，价格比按量付费更实惠，且购买时长越长，折扣越多。
   - 按量付费：属于后付费，即按小时扣费。适合短期需求，用完可立即释放实例，节省费用。

     注意：
     按量付费可转为包年包月，包年包月无法转为按量付费。
3. 设置以下参数。

参数	说明
地域	实例所在的地理位置。购买后无法更换地域。 · 请根据目标用户所在的地理位置就近选择地域，提升用户访问速度。 · 请确保Redis实例与需要连接的ECS实例或RDS实例创建于同一个地域，否则它们无法通过内网互通，只能通过外网互通，无法发挥最佳性能。
可用区	可用区是地域中的一个独立物理区域，不同可用区之间没有实质性区别。

参数	说明
网络类型	· 经典网络：传统的网络类型。 · 专有网络（推荐）：也称为VPC（Virtual Private Cloud）。VPC是一种隔离的网络环境，安全性和性能均高于传统的经典网络。 注意： · 请确保Redis实例与需要连接的ECS实例或RDS实例网络类型一致，否则它们无法通过内网互通。 · 如果Redis实例与需要连接的ECS实例或RDS实例的网络类型都是专有网络，请确保各实例在同一VPC中，否则它们无法通过内网互通。 · 经典网络中的Redis实例可以切换到专有网络，专有网络中的实例无法切换到经典网络。
虚拟交换机	虚拟交换机（VSwitch）是组成专有网络的基础网络模块。如果VPC内还没有交换机，请先创建交换机。
版本类型	Redis的引擎版本： · 2.8 · 4.0 · 5.0
架构类型	Redis实例的产品系列，详细信息请参见#unique_9。各版本之间可以相互变配，但标准版变配到集群版需注意命令使用限制，详情请参见Redis命令。
性能类型	· 标准性能：在Redis社区版的基础上进行了优化的标准版本，性能与社区版Redis相似。 · 增强性能：经过多线程性能优化的版本，性能是同规格社区版Redis的3倍左右，详细信息请参见#unique_11。
存储类型	· 高性能内存型：使用内存保存全量数据。 · 混合存储型：使用磁盘保存全量数据，将热数据存储在内存中，详细信息请参见#unique_12。
节点类型	标准版与集群版有两种节点类型： · 双副本为一主一从的双机热备架构，数据持久化保存； · 单副本为单节点缓存架构，没有数据可靠性保障。 读写分离版有三种节点类型： · 只读节点（1个）； · 只读节点（3个）； · 只读节点（5个）。

参数	说明
套餐类型	· 标准套餐：提供基础规格。 · 定制套餐：提高最大连接数和内网带宽上限的高级版本。
实例规格	每种规格都有对应的内存大小、连接数上限、带宽限制等，详情请参见#unique_13。  说明: 实例创建后会自动生成数据库元信息，占用少量的存储空间： · 标准版实例中的元信息大小约为32MB； · 集群版实例中的元信息大小约为集群节点数量*32MB。

4. 选择密码设置方式。
   · 立即设置：在下方的输入密码区域设置密码。
   · 稍后设置：创建实例后再#unique_14。
5. 设置实例名称、购买数量，如果创建包年包月实例，还需设置购买时长。
6. 单击右侧的立即购买按钮。
7. 在确认订单页，阅读《云数据库KVStore版服务协议》，确认接受后在服务协议前的选框中单击勾选。
8. 单击去开通。

 说明:
支付成功后会提示开通成功。等待1-5分钟即可在控制台看到新购买的Redis实例。

视频演示

您可以通过观看以下视频了解如何创建实例及登录实例，视频时长约4分钟。

相关API

API	说明
#unique_15	创建一个Redis实例。

3 步骤2：设置白名单

为了Redis数据库的安全稳定，在开始使用Redis实例前，您需要将访问数据库的IP地址或者IP段添加到目标实例的白名单中。正确使用白名单可以让Redis得到高级别的访问安全保护，建议您定期维护白名单。

前提条件

白名单功能需要特定版本内核的支持，若实例的内核版本不满足条件，在设置白名单时会出现提示信息，此时将小版本升级到最新即可，详细步骤请参见#unique_17。

操作步骤

1. 登录Redis管理控制台。
2. 在界面左上方的菜单栏中选择实例所在的地域 。
3. 在实例列表页，单击目标实例ID或者其右侧操作栏的管理。
4. 在实例信息页，单击左侧导航栏中的白名单设置。
5. 在白名单设置页，从以下方式中选择其一继续。
   - 如果需要自定义白名单分组名称，则创建新的白名单分组：
     a. 单击右侧的添加白名单分组。
     b. 设置分组名称。

     > 说明:
     > 分组名称长度为2~32个字符，由小写字母、数字或下划线组成，开头需为小写字母，结尾需为字母或数字。在白名单分组创建成功后，该名称将不能被修改。
   - 如果不需要创建自定义分组，单击目标白名单分组右侧的修改。
6. 在添加白名单分组或修改白名单分组对话框中，从以下方式中选择其一继续。
   - 手动修改组内白名单：
     a. 在组内白名单区域手动输入允许访问Redis实例的IP地址或者IP段。

     图 3-1: 手动修改组内白名单

     > 说明:
     > - 设置`0.0.0.0/0`表示允许所有地址访问。

- 仅设置127.0.0.1表示禁止所有地址访问。
- 设置IP段表示该网段内的IP地址都可以访问该Redis实例，如10.10.10.0/24。
- 多个IP用英文逗号隔开，逗号前后不加空格。
- 每个白名单分组可以设置1000个IP。

b. 单击确定。

· 加载当前阿里云账号下ECS实例的内网IP：

a. 单击加载ECS内网IP。

图 3-2: 加载ECS内网IP

b. 选择目标ECS的内网IP。

图 3-3: 选择ECS内网IP

说明:
您可以在ECS内网IP列表上方的检索栏中通过ECS实例的名称、ID或IP进行模糊搜索。

c. 单击确定。

相关API

API	说明
#unique_18	调用DescribeSecurityIps查询允许访问Redis实例的IP名单。
#unique_19	调用ModifySecurityIps设置Redis实例的IP白名单。

4 步骤3：连接实例

4.1 DMS登录云数据库

数据管理DMS（Data Management）是一款访问管理云端数据的Web服务，支持Redis、MySQL、SQL Server、PostgreSQL和MongoDB等数据源。DMS提供了数据管理、对象管理、数据流转和实例管理四部分功能。

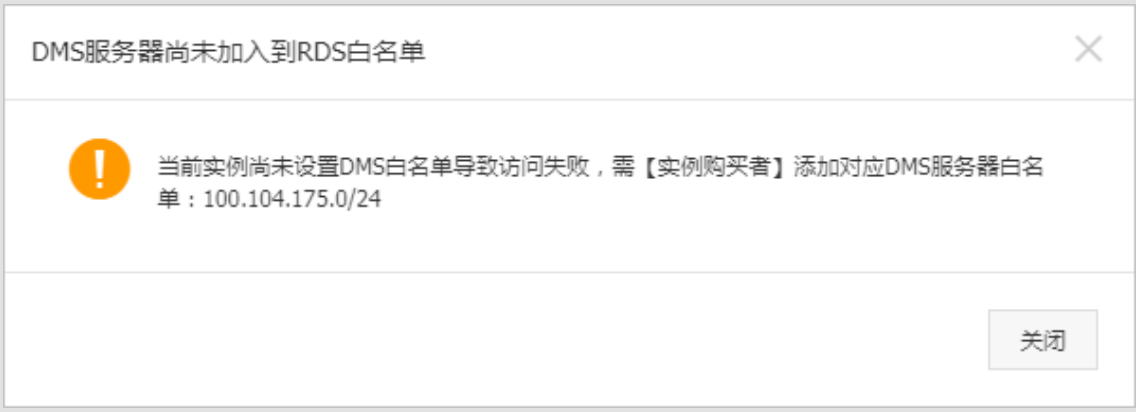
您可以通过以下两种方式使用DMS登录数据库。

> 说明:
> 如果出现连接问题，请参见连接问题排查与解决。

从Redis管理控制台登录

操作步骤

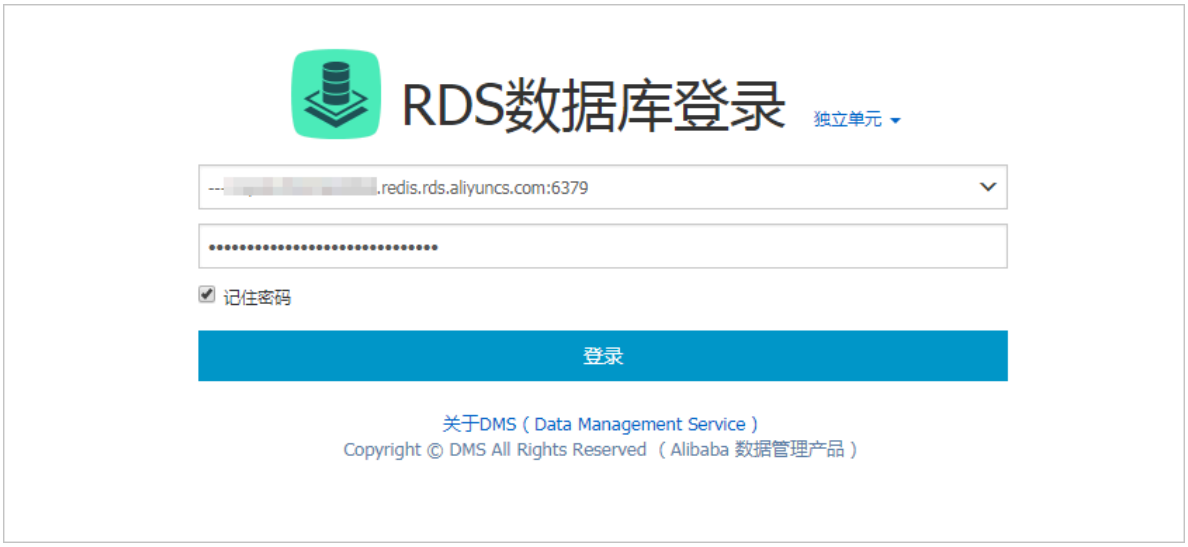
1. 登录Redis管理控制台。
2. 在界面左上方的菜单栏中选择实例所在的地域 。
3. 在实例列表中，单击目标实例右侧操作栏的管理，或者单击实例ID。
4. 在实例信息页，单击右上方的登录数据库。

> 说明:
> - 首次使用DMS登陆Redis会弹出提示，请您将DMS服务器的IP地址加入Redis的白名单。
>
> 
>
>
> - 通过该方式打开DMS，系统会自动填写登录页面中的数据库连接地址，您只要输入密码即可。
> - 如果实例开启了免密访问功能，则在同一VPC内无需提供密码信息即可连接数据库。

从DMS管理控制台登录

操作步骤

1. 登录DMS管理控制台。
2. 输入要登录的数据库连接地址和密码，单击登录，如下图所示。



说明：

- 连接信息可以在Redis管理控制台的实例信息页获取。
- 经典网络及VPC网络的实例都支持DMS。使用DMS登录VPC网络中的实例时，系统需要申请一条特殊通道，所以该类实例在第一次登录时，会有短暂的等待时间。
- 每次登录DMS都会触发SCAN命令扫描Redis数据，因此会被动逐出部分过期数据机制。

更多的DMS相关信息请参见DMS官方文档。

4.2 Redis客户端连接

您可以使用多种语言的客户端连接阿里云Redis。

由于云数据库Redis提供的数据库服务与原生的数据库服务完全兼容，连接数据库的方式也基本类似。任何兼容Redis协议的客户端都可以访问云数据库Redis版服务，您可以根据自身应用特点选用任何Redis客户端。

注意：

- 如果同一VPC内的实例开启了免密访问功能，则无需提供密码信息，即可连接数据库。
- 如果连接遇到问题，请参见#unique_26。

- 在使用客户端连接Redis实例前，您需要先将ECS实例的内网IP地址或本地主机的外网IP地址加入Redis的白名单。

Redis的客户端请参考http://redis.io/clients。

- Jedis客户端
- phpredis客户端
- redis-py客户端
- C/C++客户端
- .net客户端
- node-redis客户端
- C#客户端StackExchange.Redis

Jedis客户端

Jedis客户端访问云数据库Redis版服务，有以下两种方法：

- Jedis单链接
- JedisPool连接池连接

操作步骤如下：

1. 下载并安装Jedis客户端：单击下载地址。
2. Jedis单连接示例

   a. 打开Eclipse客户端，创建一个Project，输入如下代码段：

```
import redis.clients.jedis.Jedis;
public class jedistest {
public static void main(String[] args) {
try {
     String host = "xx.kvstore.aliyuncs.com";//控制台显示访问地址
     int port = 6379;
     Jedis jedis = new Jedis(host, port);
     //鉴权信息
     jedis.auth("password");//password
     String key = "redis";
     String value = "aliyun-redis";
     //select db默认为0
     jedis.select(1);
     //set一个key
     jedis.set(key, value);
     System.out.println("Set Key " + key + " Value: " + value);
     //get 设置进去的key
     String getvalue = jedis.get(key);
     System.out.println("Get Key " + key + " ReturnValue: " +
getvalue);
     jedis.quit();
     jedis.close();
}
catch (Exception e) {
 e.printStackTrace();
```

```
 }
}
}
```

b. 运行上述Project，在Eclipse的控制台输出如下运行结果则表示您已成功连接云数据库Redis。

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

接下来您就可以通过自己的本地客户端Jedis操作您的云数据库Redis。您也可以通过JedisPool连接池来连接您的云数据库Redis。

3. JedisPool连接池示例

a. 打开Eclipse客户端，创建一个Project，配置pom文件，具体配置如下所示：

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

b. 在project中添加如下应用：

```
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. 如果您的Jedis客户端版本是Jedis-2.7.2，在Project中输入如下代码：

```
JedisPoolConfig config = new JedisPoolConfig();
//最大空闲连接数，应用自己评估，不要超过ApsaraDB for Redis每个实例最大的连接数
config.setMaxIdle(200);
//最大连接数，应用自己评估，不要超过ApsaraDB for Redis每个实例最大的连接数
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "密码";
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. 如果您的Jedis客户端版本是Jedis-2.6、Jedis-2.5，在Project中输入如下代码：

```
JedisPoolConfig config = new JedisPoolConfig();
//最大空闲连接数，应用自己评估，不要超过ApsaraDB for Redis每个实例最大的连接数
config.setMaxIdle(200);
//最大连接数，应用自己评估，不要超过ApsaraDB for Redis每个实例最大的连接数
config.setMaxTotal(300);
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
String host = "*.aliyuncs.com";
String password = "密码";
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. 运行上述Project，在Eclipse的控制台输出如下运行结果则表示您已成功连接云数据库Redis。

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

接下来您就可以通过自己的本地客户端Jedis操作您的云数据库Redis。

phpredis客户端

操作步骤如下所示：

1. 下载并安装phpredis客户端：单击下载地址。

2. 在任何一款可以编辑php的编辑器中输入如下代码：

```
<?php
 /* 这里替换为连接的实例host和port */
 $host = "localhost";
 $port = 6379;
 /* 这里替换为实例id和实例password */
 $user = "test_username";
 $pwd = "test_password";
 $redis = new Redis();
 if ($redis->connect($host, $port) == false) {
         die($redis->getLastError());
   }
 if ($redis->auth($pwd) == false) {
         die($redis->getLastError());
  }
  /* 认证后就可以进行数据库操作，详情文档参考https://github.com/phpredis/
phpredis */
 if ($redis->set("foo", "bar") == false) {
         die($redis->getLastError());
 }
 $value = $redis->get("foo");
 echo $value;
 ?>
```

3. 执行上述代码，您就可以通过自己的本地客户端phpredis访问您的云数据库Redis，详情文档参考https://github.com/phpredis/phpredis 。

redis-py客户端

操作步骤如下：

1. 下载并安装redis-py客户端：单击下载地址。
2. 在任何一款可以编辑Python的编辑器中输入如下代码，即可建立连接通过本地客户端redis-py进行数据库操作。

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
import redis
#这里替换为连接的实例host和port
host = 'localhost'
port = 6379
#这里替换为实例password
pwd = 'test_password'
r = redis.StrictRedis(host=host, port=port, password=pwd)
#连接建立后就可以进行数据库操作，详情文档参考https://github.com/andymccurdy/
redis-py
r.set('foo', 'bar');
print r.get('foo')
```

C/C++客户端

操作步骤如下所示：

1. 下载并编译安装C客户端，编译安装代码如下所示：

```
      git clone https://github.com/redis/hiredis.git
```

```
cd hiredis
make
sudo make install
```

2. 在C/C++编辑器中编写如下代码：

```
#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. 编译上述代码。

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

4. 测试运行。

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

至此完成通过C/C++客户端连接云数据库Redis。

.net客户端

操作步骤如下所示：

1. 下载并使用.net客户端。

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

2. 在.net 客户端中新建.net项目。
3. 添加客户端引用，引用文件在库文件的ServiceStack.Redis/lib/tests中。
4. 在新建的.net项目中输入如下代码来连接云数据库Redis。详细的接口用法请参见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";/*访问host地址*/
          string password = "password";/*密码*/
          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://密码@访问地址:端
口*/
 };
 RedisConfig.VerifyMasterConnections = false;//需要设置
 PooledRedisClientManager redisPoolManager = new PooledRedi
sClientManager(10/*连接池个数*/, 10/*连接池超时时间*/, testReadWr
iteHosts);
 for (int i = 0; i < 100; i++){
          IRedisClient redisClient = redisPoolManager.GetClient
();//获取连接
          RedisNativeClient redisNativeClient = (RedisNativeClient)
redisClient;
          redisNativeClient.Client = null;//ApsaraDB for Redis不支持
client setname所以这里需要显示的把client对象置为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)
 {
          //单链接模式
          RedisClientTest();
          //连接池模式
          RedisPoolClientTest();
 }
 }
```

```
}
```

node-redis客户端

操作步骤如下所示：

1. 下载并安装node-redis。

```
npm install hiredis redis
```

2. 在node-redis客户端中输入如下代码并执行以此连接云数据Redis版。

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

说明：

port为端口，默认为6379；host为连接地址。例如：

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

3. 使用云数据Redis版。

```
  // 写入数据
client.set("key", "OK");
// 获取数据，返回String
client.get("key", function (err, reply) {
console.log(reply.toString()); // print `OK`
});
// 如果传入一个Buffer，返回也是一个Buffer
client.get(new Buffer("key"), function (err, reply) {
console.log(reply.toString()); // print `<Buffer 4f 4b>`
});
client.quit();
```

C#客户端StackExchange.Redis

操作步骤如下所示：

1. 下载并安装StackExchange.Redis。
2. 添加引用。

```
using StackExchange.Redis;
```

3. 初始化ConnectionMultiplexer。

   ConnectionMultiplexer是StackExchange.Redis的核心，它被整个应用程序共享和重用，应该设置为单例，它的初始化如下：

```
// redis config
```

```
 private static ConfigurationOptions configurationOptions =
ConfigurationOptions.Parse("127.0.0.1:6379,password=xxx,connectTim
eout=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;
 }
```

说明:

ConfigurationOptions包含很多选项，例如keepAlive、connectRetry、name，具体可以参考StackExchange.Redis.ConfigurationOptions。

4. **GetDatabase()返回的对象是轻量级的，每次用的时候从ConnectionMultiplexer对象中获取即可。**

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

5. **下面给出5种数据结构的demo，它们的API和原生略有不同，分别用String、Hash、List、Set、SortedSet开头代表5种数据结构。**

   - **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
 );
```

- 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);
//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 redis-cli连接

您可以使用原生工具redis-cli来连接阿里云Redis。

redis-cli连接简介

redis-cli是原生Redis自带的命令行工具，可以帮助您通过简单的命令连接Redis实例，进行数据管理。

使用redis-cli，您可以在阿里云ECS实例上的Linux系统中连接云数据库Redis版实例，或者在本地主机上通过外网访问Redis实例。通过阿里云内网访问Redis实例能够提供更高的安全和性能保障，您可以在ECS上使用redis-cli与同一VPC内的Redis实例或者同地域的经典网络Redis实例建立连接；如果您的场景需要在本地主机上从外网访问Redis实例，请先按照#unique_28的说明申请外网连接地址，再结合本文的连接方法，使用redis-cli连接Redis实例。

redis-cli安装方法

在Linux环境中安装原生Redis即可使用redis-cli。具体步骤请参见Redis社区版官网。

前提条件

内网连接

- ECS实例与Redis实例的网络类型同为VPC时，二者需在同一地域的同一VPC中。
- ECS实例与Redis实例的网络类型同为经典网络时，二者需在同一地域。
- Redis实例的白名单中加入了ECS实例的内网IP地址。
- ECS中已安装原生Redis。

外网连接

- Redis实例申请了外网连接地址，申请方式参见#unique_28。
- Redis实例的白名单中加入了本地主机的外网IP地址。
- 本地主机的系统为Linux。

- 本地主机已安装原生Redis。

注意事项

- 如果通过内网地址访问Redis实例并开启了VPC免密，连接时无需密码验证。
- 如果通过外网地址访问Redis实例并开启了VPC免密，连接时仍然需要密码验证。
- 如果连接遇到问题，请参见#unique_26。

连接方法

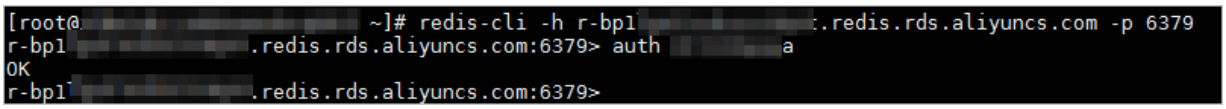
在Linux命令行使用如下命令连接Redis实例：

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

表 4-1: 选项说明

选项	说明
-h	Redis实例的连接地址。 · 内网连接：使用内网连接地址； · 外网连接：使用外网连接地址。
-p	Redis实例的服务端口。 · 内网连接：默认为6379，不可修改； · 外网连接：默认为6379，可自定义，修改方式参见#unique_31。
-a	Redis实例的连接密码。为了提高安全性，您也可以不设置该选项，在执行连接命令后再输入`auth <password>`密码验证，如下方图片所示。

图 4-1: 连接示例



视频演示

您可以通过观看以下视频来了解如何通过redis-cli连接Redis实例，视频时长约2分钟。

4.4 外网连接

Redis外网连接地址，又称公网连接地址，是一个可以通过因特网直接访问的地址。您可以通过外网连接地址从外网访问云数据库Redis版实例。相对于使用内网连接地址连接Redis实例，使用

外网连接地址会提高网络耗时，影响Redis服务的性能。在生产环境中，建议您通过内网地址连接Redis，确保Redis服务的高性能。

前提条件

- 请确保Redis实例的白名单中设置了ECS实例或本地主机的外网IP地址。设置方式请参见步骤2：设置白名单。
- 2.8或5.0版本的Redis实例在#unique_33后无法申请外网连接地址，如需使用外网连接请先关闭免密访问再申请外网连接地址。

> 说明:
> 4.0版本的Redis实例可以在开启免密访问后申请外网连接地址，此时使用内网地址访问Redis实例无需密码，使用外网地址访问仍需要密码。

适用场景

- 本地访问：通过本地主机连接阿里云Redis实例。
- 跨账号访问：通过ECS连接不同阿里云账号下的Redis实例。
- 跨地域访问：同一阿里云账号下的ECS实例与Redis实例需要互连，但二者属于不同地域。
- 跨VPC访问：同一阿里云账号下的ECS实例与Redis实例需要互连，二者属于同地域但VPC不同。
- 跨网络类型访问：同一阿里云账号下的ECS实例与Redis实例需要互连，二者属于相同地域，但网络类型不同。

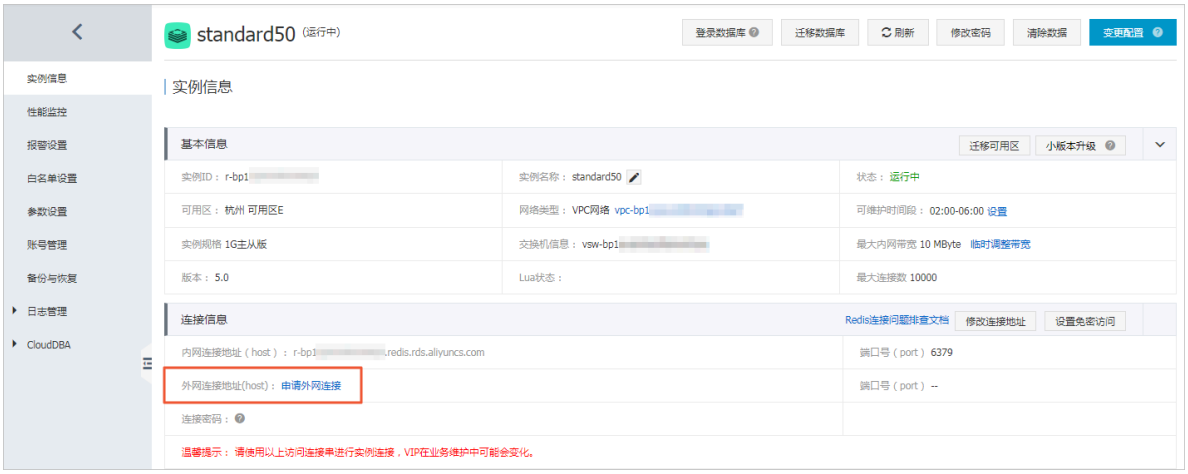
费用

Redis外网连接功能及外网流量暂不收取费用。

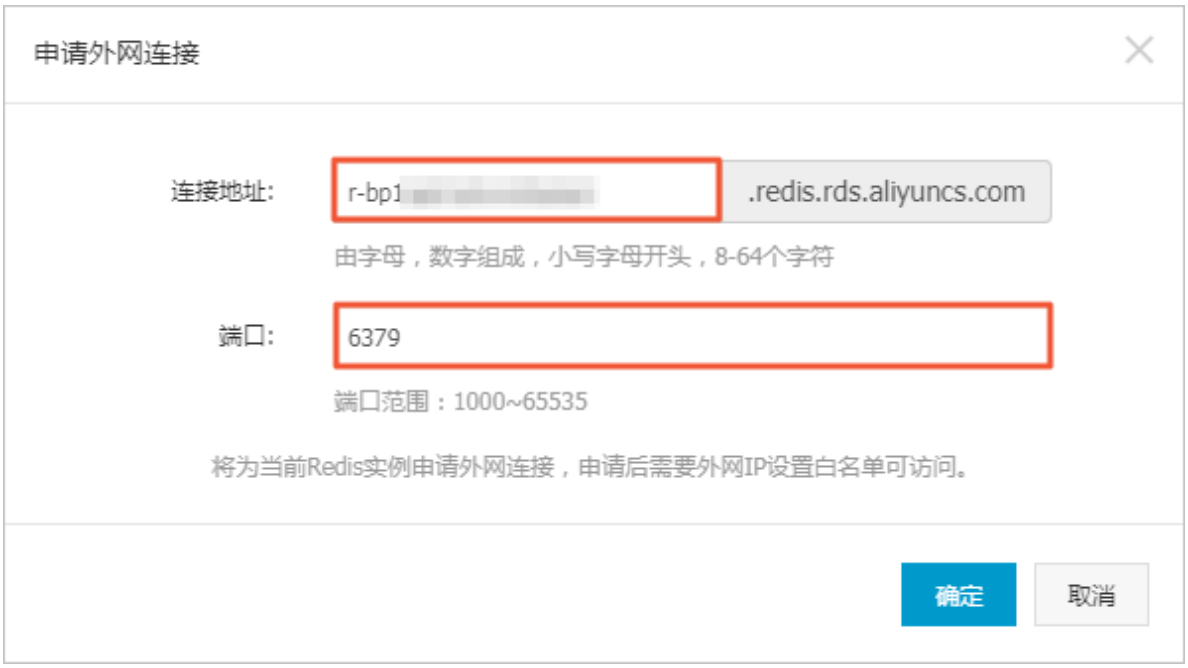
获取外网连接地址

1. 登录Redis管理控制台。
2. 在界面左上方阿里云图标的右侧选择实例所在的地域 。
3. 在实例列表页，单击目标实例ID或者其右侧操作栏的管理。

4. 在实例信息页，单击连接信息区域的申请外网连接。

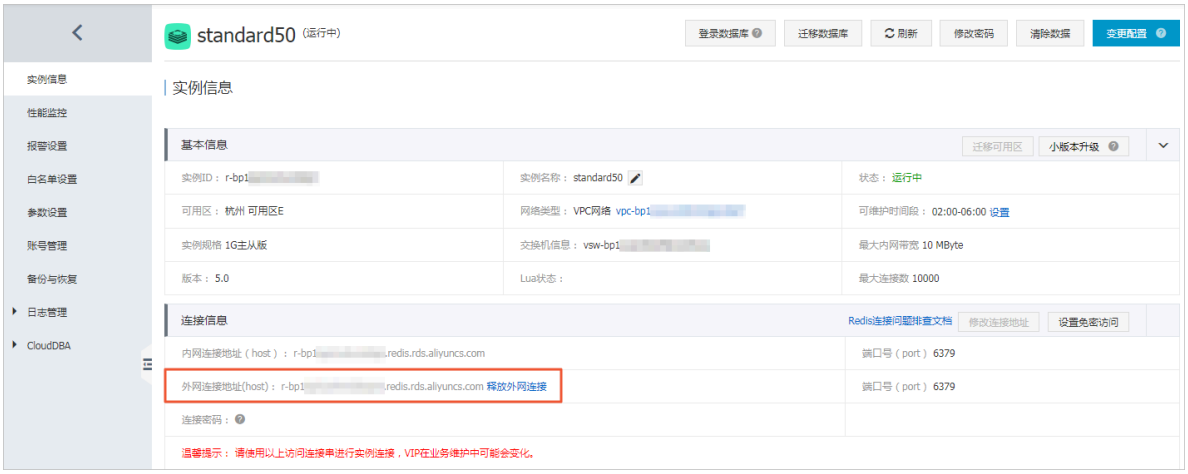


5. 在申请外网连接对话框输入自定义的连接地址和端口，或保留默认值，之后单击确定。



6. 在实例信息页，查看连接信息的外网连接地址。

图 4-2: Redis外网连接地址



说明:

如果不再需要外网连接地址，单击外网连接地址右侧的释放外网连接按钮释放该地址。

使用外网地址连接实例

您可以使用redis-cli或各语言的Redis客户端等工具连接Redis实例，连接方式请参见如下文档：

- DMS登录云数据库
- #unique_34
- #unique_35

外网连接失败的解决方法

- 请确认使用的地址是Redis实例的外网连接地址而不是内网连接地址，外网连接地址的位置请参见图 4-2: Redis外网连接地址。
- 请检查Redis白名单中是否添加了客户端所在主机的外网IP地址。
- 内网连接问题请参见#unique_36。

5 Redis管理控制台

Redis管理控制台是用于管理Redis实例的Web应用程序，管理Redis实例所需的基础操作都可以在控制台上完成。您可以通过本文档了解控制台的使用方法。

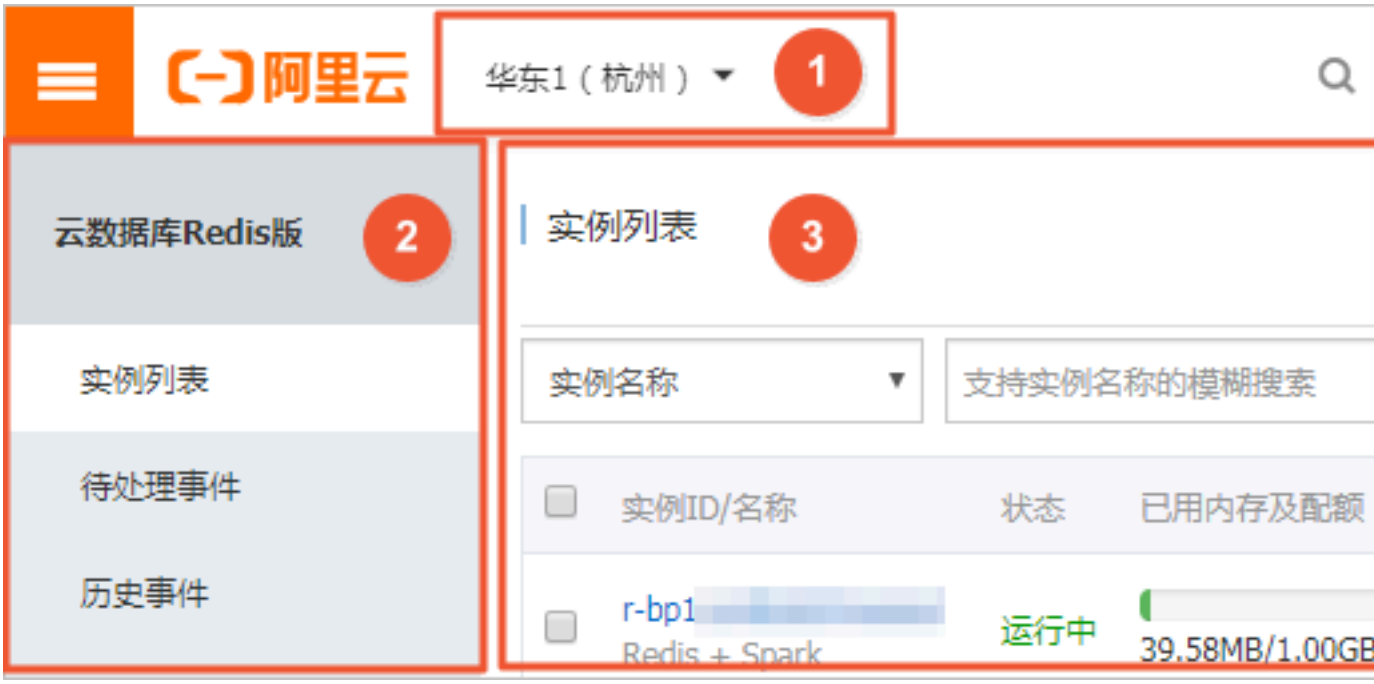
Redis管理控制台是阿里云管理控制台的一部分，关于控制台的通用设置和基本操作请参见使用阿里云管理控制台。本文将介绍Redis控制台的通用界面，若有差异，请以控制台实际界面为准。

控制台首页

登录Redis管理控制台即进入控制台首页。对于Redis所有类型的实例而言，控制台首页的界面信息都是相同的。

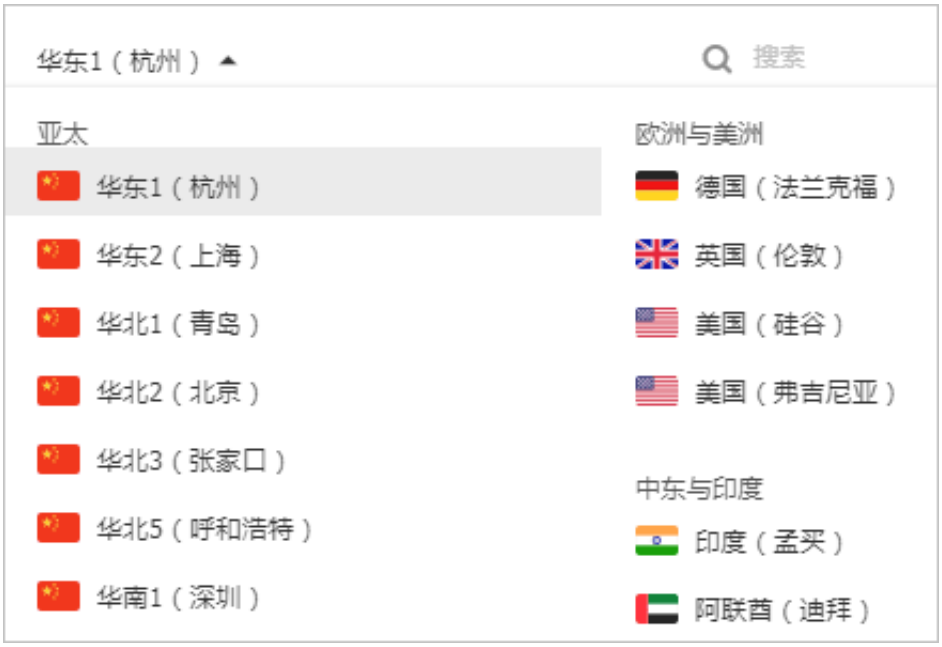
> **说明:**
> 若没有阿里云账号，请先注册。

图 5-1: Redis管理控制台首页



控制台首页主要功能区介绍如下。

- 区域1为地域选择菜单，单击或光标悬浮于该区域即展示区域列表。单击目标地域即可切换到该地域的实例列表。



- 区域2为首页左侧导航栏。登录Redis管理控制台后默认进入实例列表页。
  - 实例列表中展示本地域的实例资源列表。详情请参见下文对区域3的说明。
- 区域3为实例列表页。实例列表页面中会展示实例ID、状态、已用内存及配额、可用区、版本、实例规格、创建时间、付费方式、网络类型等信息。

> **说明:**
>
> 已用内存及配额是由底层系统根据采集信息进行离线汇总得到的结果，存在10分钟左右的延迟，与当前时间的实际值可能存在差别。
>
> 如果需要查看实时信息，建议登录DMS进行查看，详细步骤请参见#unique_38。

控制台常用功能导航

- 性能监控
- 报警设置
- 白名单设置
- 参数设置
- 账号管理
- 备份与恢复
- 日志管理
- 缓存分析
- DMS for Redis

6 使用限制

项目	说明
List数据类型	没有List个数限制，单个元素最大值为512 MB，推荐list的元素个数小于8192，value最大长度不超过1 MB。
Set数据类型	没有set个数限制，单个元素最大值为512 MB，推荐set的元素个数小于 8192， value最大长度不超过 1 MB。
Sorted set数据类型	没有sorted set个数限制，单个元素最大值为512 MB，推荐sorted set的元素个数小8192，value最大长度不超过1 MB。
Hash数据类型	没有field个数限制，单个元素最大值为512 MB，推荐元素个数小于8192， value最大长度不超过1 MB。
DB数限制	每个实例支持256个DB。 说明: · 所有DB存储的数据总量受限于实例的内存大小； · 单个DB占用内存按照使用情况自动分配，上限为实例内存（比如db0占用全部内存而其它db无数据）。
Redis 命令支持	详情请参见#unique_47。
监控报警	云数据库Redis版未提供容量告警，需要用户到云监控中进行配置。配置方法请参见文档。 建议设置好以下监控的报警：实例故障、实例主备切换、已使用连接百分比、操作失败数、已用容量百分比、写入带宽使用率、读取带宽使用率。
数据过期删除策略	· 主动过期，系统后台会周期性的检测，发现已过期的key时，会将其删除。 · 被动过期，当用户访问某个key时，如果该key已经过期，则将其删除。
空闲连接回收机制	服务端不主动回收Redis空闲连接，由用户管理。
数据持久化策略	采用AOF_FSYNC_EVERYSEC方式，每秒fsync。

7 Redis命令

本文介绍云数据库Redis 2.8版本和4.0版本支持的Redis命令，以及暂未开放或受限制的Redis命令。云数据库Redis版兼容Redis 3.0版本，支持Redis 3.0的Geo命令。

支持的Redis命令

表 7-1: Redis命令支持表1

Keys（键）	String（字符串）	Hash（哈希表）	List（列表）	Set（集合）	SortedSet（有序集合）
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（键）	String（字符串）	Hash（哈希表）	List（列表）	Set（集合）	SortedSet（有序集合）
TYPE	PSETEX		RPUSHX		ZSCAN
SCAN	SET				ZRANGEBYLEX
OBJECT	SETBIT				ZLEXCOUNT
	SETEX				ZREMRANGEBYLEX
	SETNX				
	SETRANGE				
	STRLEN				

表 7-2: Redis命令支持表2

HyperLogLog	Pub/Sub（发布/订阅）	Transaction（事务）	Connection（连接）	Server（服务器）	Scripting（脚本）	Geo（地理位置）
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		

HyperLogLog	Pub/Sub（发布/订阅）	Transaction（事务）	Connection（连接）	Server（服务器）	Scripting（脚本）	Geo（地理位置）
				CONFIG GET		
				MONITOR		
				SLOWLOG		

说明:

- 在Redis集群实例中，client list命令列出所有连接到该proxy的user connection。其中，id、age、idle、addr、fd、name、db、multi、omem、cmd字段和Redis内核表达的意思一样。sub、psub在proxy层没有区分，要么都为1，要么都为0。qbuf、qbuf-free、obl、oll字段目前没有意义。
- 在Redis集群实例中，client kill命令目前支持两种形式：`client kill ip:port`和`client kill addr ip:port`。

4.0版本新增Redis命令

表 7-3: 4.0 Redis命令

Keys（键）	Server（服务器）
UNLINK	SWAPDB
	MEMORY

4.0更新的Redis命令

FLUSHALL/FLUSHDB新增选项ASYNC。

说明:

添加ASYNC选项后FLUSHALL或FLUSHDB的操作将在新的线程中异步进行，不会阻塞服务。更多Redis 4.0相关特性请参见Redis 4.0命令及新特性介绍。

暂未开放的Redis命令

Keys（键）	Server（服务器）
MIGRATE	BGREWRITEAOF
	BGSAVE

Keys（键）	Server（服务器）
	CONFIG REWRITE
	CONFIG SET
	CONFIG RESETSTAT
	COMMAND
	COMMAND COUNT
	COMMAND GETKEYS
	COMMAND INFO
	DEBUG OBJECT
	DEBUG SEGFAULT
	LASTSAVE
	ROLE
	SAVE
	SHUTDOWN
	SLAVEOF
	SYNC

集群实例受限制的Redis命令

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

说明:

- 集群实例受限的Redis命令只支持所操作key均分布在单个hash slot中的场景，没有实现多个hash slot数据的合并功能，因此需要用hash tag的方式确保要操作的key均分布在一个hash slot中。

  比如有key1，aakey，abkey3，那么我们在存储的时候需要用{key}1，aa{key}，ab{key}3的方式存储，这样调用受限命令时才能生效。具体关于hash tag的用法请参见Redis官方文档：http://redis.io/topics/cluster-spec。
- 事务之前没有使用watch命令且事务中都是单key的命令场景，不再要求所有key必须在同一个slot中，使用方式和直连redis完全一致。其他场景要求事务中所有命令的所有key必须在同一个slot中。
  - 多key命令包括：DEL、SORT、MGET、MSET、BITOP、EXISTS、MSETNX、RENAME、 RENAMENX、BLPOP、BRPOP、RPOPLPUSH、BRPOPLPUSH、SMOVE、SUNION、SINTER、SDIFF、SUNIONSTORE、SINTERSTORE、SDIFFSTORE、ZUNIONSTORE、ZINTERSTORE、 PFMERGE、PFCOUNT。
  - 不允许在事务中使用的命令包括：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、CLIENT。

Lua使用限制

Lua脚本放开限制，标准版-双节点、标准版-单节点支持用户直接调用。

集群版本条件性支持：

- 所有key都应该由KEYS数组来传递，redis.call/pcall中调用的redis命令，key的位置必须是KEYS array（不能使用Lua变量替换KEYS），否则直接返回错误信息，"`-ERR bad lua script for redis cluster, all the keys that the script uses should be passed using the KEYS array\r\n`"。
- 所有key必须在1个slot上，否则返回错误信息，"`-ERR eval/evalsha command keys must be in same slot\r\n`"。
- 调用必须要带有key，否则直接返回错误信息，"`-ERR for redis cluster, eval/evalsha number of keys can't be negative or zero\r\n`"。

自研的集群实例Redis命令

- info key命令：查询key所属的slot和db。Redis原生的info命令中最多可以带一个可选的section (`info [section]`)。目前云数据库Redis版的集群实例，部分命令限制所有key必

须在同一个slot中，`info key`命令方便用户查询某些key是否在同一个slot或db节点中。用法如下：

```
127.0.0.1:6379> info key test_key
slot:15118 node_index:0
```

注意：

- 线上旧版本可能出现`info key`显示出来的`node index`和实例拓扑图的`node index`不一致，最新版本已经修复。
- `info key`显示的node是指集群规格下后端的物理节点，和select命令中的db不是同一个概念。

- iinfo命令：用法类似于info，用于在指定的Redis节点上执行info命令。用法如下：

```
iinfo db_idx [section]
```

其中，db_idx的范围是[0, nodecount]，nodecount可以通过info命令获取，section的用法与官方info命令中的section一致。要了解某个Redis节点的info可以使用iinfo命令或者从控制台上查看实例拓扑图，详情请参见 如何查看Redis集群子实例内存。

- riinfo命令：和iinfo命令类似，但只能在读写分离的模式下使用。用法中增加了一个readonly slave的idx，用于指定在第几个readonly slave上执行info命令。在读写分离集群中可以用来在指定readonly slave上执行info命令。如果在非读写分离集群中使用，会返回错误。用法如下：

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

- iscan命令：在集群模式下可以在指定的db节点上执行scan命令。在scan命令的基础上扩展了一个参数用于指定db_idx， db_idx的范围是[0, nodecount]，nodecount可以通过info命令获取或者从控制台上查看实例拓扑图。用法如下：

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

- imonitor命令：和iinfo、 iscan类似，在monitor的基础上新增一个参数指定monitor执行的db_idx，db_idx的范围是[0, nodecount)， nodecount可以通过info命令获取或者从控制台上查看实例拓扑图。用法如下：

```
imonitor db_idx
```

- rimonitor命令：和riinfo类似，用于读写分离场景下，在指定的shard里的指定只读从库上执行monitor命令。用法如下：

```
rimonitor db_idx ro_slave_idx
```

说明：
imonitor和rimonitor请用telnet连接后执行，如需退出imonitor/rimonitor，请使用`quit`命令。

说明

- 关于Redis命令的详细信息，请参见官方文档。
- 如果在集群规格实例上使用已经支持的命令仍然提示`unkown command`，请在控制台升级小版本。
- 云数据库Redis版集群实例最新的命令支持详情，请参见云栖社区说明。