ALIBABA CLOUD

Alibaba Cloud

ApsaraDB for Redis

Data migration and synchronization

Document Version: 20200928

Alibaba Cloud

Legal disclaimer

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

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

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

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

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

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

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

Document conventions

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

Table of Contents

1.Migrate data

1.1. Overview

You can migrate your workloads between on-premises Redis databases and ApsaraDB for Redis without service disruption.

ApsaraDB for Redis provides multiple migration solutions to help you migrate data in different scenarios. You can use the following methods to manage migrations:

- Use Alibaba Cloud Data Transmission Service (DTS) to migrate full and incremental data in Redis databases.
- Use append-only files (AOF) or RDB files to migrate data.
- Use redis-shake or other tools to migrate data.

The following table helps you navigate to the documents that describe migration solutions in different scenarios.

Migration solutions

Scenario	Solution
Migrate data from an on-premises Redis database to ApsaraDB for Redis	Use the redis-shake tool to migrate the data of an on-premises Codis or Redis cluster to the cloud
	Use redis-shake to migrate data
	Use DTS to migrate data
	Use the redis-shake tool to migrate data from an RDB file
	Use AOF files to migrate data
Migrate data from ApsaraDB for Redis to an on-premises Redis database	Migrate backup sets
Migrate data between ApsaraDB for Redis instances	Use redis-shake to migrate data
Migrate data from a third-party database to ApsaraDB for Redis	Migrate data from Amazon ElastiCache for Redis to ApsaraDB for Redis
	Migrate data from SSDB to ApsaraDB for Redis
	Migrate data from Google Cloud Memorystore for Redis to ApsaraDB for Redis

After the data is migrated, you can use the redis-full-check tool to check whether the data is consistent between a source database and a destination database. For more information, see Verify data after migration.

1.2. Migrate data from on-premises Redis to ApsaraDB for Redis

1.2.1. Migrate data from a user-created Redis database to an ApsaraDB for Redis instance

This topic describes how to migrate data from a user-created Redis database to an ApsaraDB for Redis instance by using Data Transmission Service (DTS). DTS supports full data migration and incremental data migration. When you migrate data from a user-created Redis database to Alibaba Cloud, you can select the two migration types to ensure service continuity.

Prerequisites

- The version of the user-created Redis database is 2.8, 3.0, 3.2, 4.0, or 5.0.
- The user-created Redis database uses the standalone architecture rather than the cluster architecture.

  > **Note** If the user-created Redis database uses the cluster architecture, you can migrate data by using the data synchronization feature. For more information, see Synchronize data from a user-created Redis cluster to an ApsaraDB for Redis cluster instance.

- The `PSYNC` or `SYNC` command can be run on the user-created Redis database.
- The available storage space of the destination ApsaraDB for Redis database is larger than the total size of the data in the user-created Redis database.

Precautions

- DTS uses resources of the source and destination databases during full data migration. This may increase the load of the database server. If the data volume is large or the specification is low, database services may become unavailable. Before you migrate data, evaluate the performance of the source and destination databases. We recommend that you migrate data during off-peak hours.
- If the data eviction policy (`maxmemory-policy`) of the destination database is not set to `noeviction` , the data between the source and destination databases may become inconsistent. For more information about the data eviction policy, see How does ApsaraDB for Redis evict data by default?
- If you use the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination database. During incremental data migration, the destination database does not explicitly return the execution results of Lua scripts.
- When calling the `PSYNC` or `SYNC` command to transmit data of the LIST type, DTS does not perform the `flush` operation on the existing data. Therefore, duplicate data may exist in the destination database.
- DTS automatically resumes a failed data migration task. Before you switch your workloads to

the destination instance, stop or release the data migration task. Otherwise, the data in the source database will overwrite the data in the destination instance after the task is resumed.

Billing

Migration type	Instance configurations	Internet traffic
Schema migration and full data migration	Free of charge.	Charged only when data is migrated from Alibaba Cloud over the Internet. For more information, see Pricing.
Incremental data migration	Charged. For more information, see Pricing.	

Migration types

- Full data migration

  DTS migrates historical data of the required objects from the user-created Redis database to the destination ApsaraDB for Redis instance.

  > **Note** To ensure data consistency, do not write data into the user-created Redis database during full data migration.

- Incremental data migration

  After full data migration is complete, DTS synchronizes incremental data from the user-created Redis database to the destination ApsaraDB for Redis instance. Incremental data migration allows you to ensure service continuity when you migrate data from a user-created Redis database to Alibaba Cloud.

Operations that can be synchronized during incremental data migration

- APPEND
- BITOP, BLPOP, BRPOP, and BRPOPLPUSH
- DECR, DECRBY, and DEL
- EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT
- FLUSHALL and FLUSHDB
- GEOADD and GETSET
- HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX
- INCR, INCRBY, and INCRBYFLOAT
- LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM
- MOVE, MSET, MSETNX, and MULTI
- PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, PSETEX, and PUBLISH
- RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX
- SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE
- ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

Preparations before incremental data migration

To ensure that incremental data migration tasks can run as expected, we recommend that you remove the limit on the replication output buffer. This topic uses a server that runs on Linux as an example.

> **Note** Skip this step if you perform only full data migration.

1. Use the redis-cli program to connect to the user-created Redis database.

   > **Note** You can use the redis-cli program after you install the Redis client. For more information, see Redis community official website.

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

   > **Note**
   > - <host>: the endpoint that is used to connect to the user-created Redis database. You can use 127.0.0.1 in this example.
   > - <port>: the service port number of the user-created Redis database. The default port number is 6379.
   > - <password>: the password for the user-created Redis database.

   Example

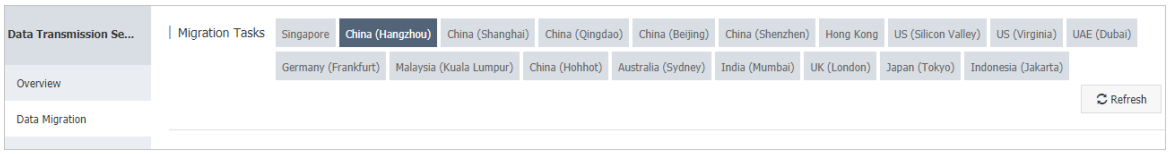
   ```
   redis-cli -h 127.0.0.1 -p 6379 -a Test123456
   ```

2. Run the following command to remove the limit on the replication output buffer:

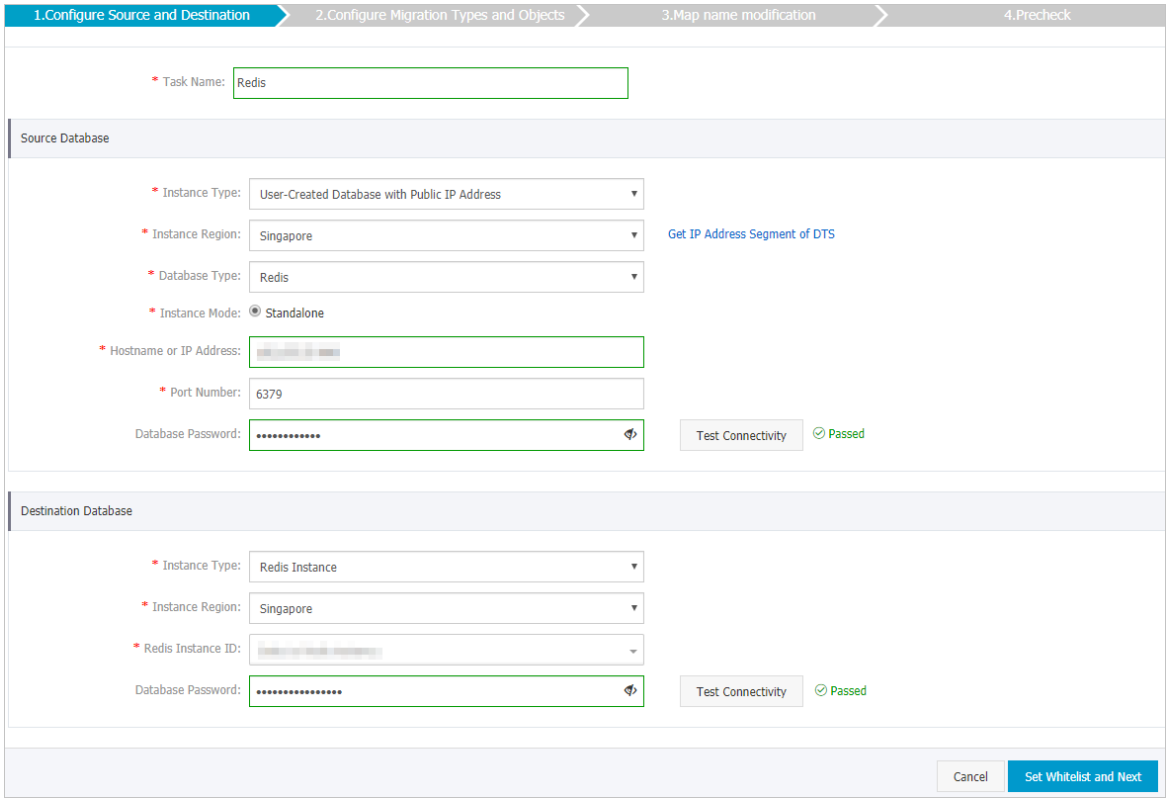
   ```
   config set client-output-buffer-limit 'slave 0 0 0'
   ```

Procedure

1. Log on to the DTS console.
2. In the left-side navigation pane, click **Data Migration**.
3. At the top of the **Migration Tasks** page, select the region where the destination RDS instance resides.



4. In the upper-right corner of the page, click **Create Migration Task**.
5. Configure the source and destination databases for the data migration task.



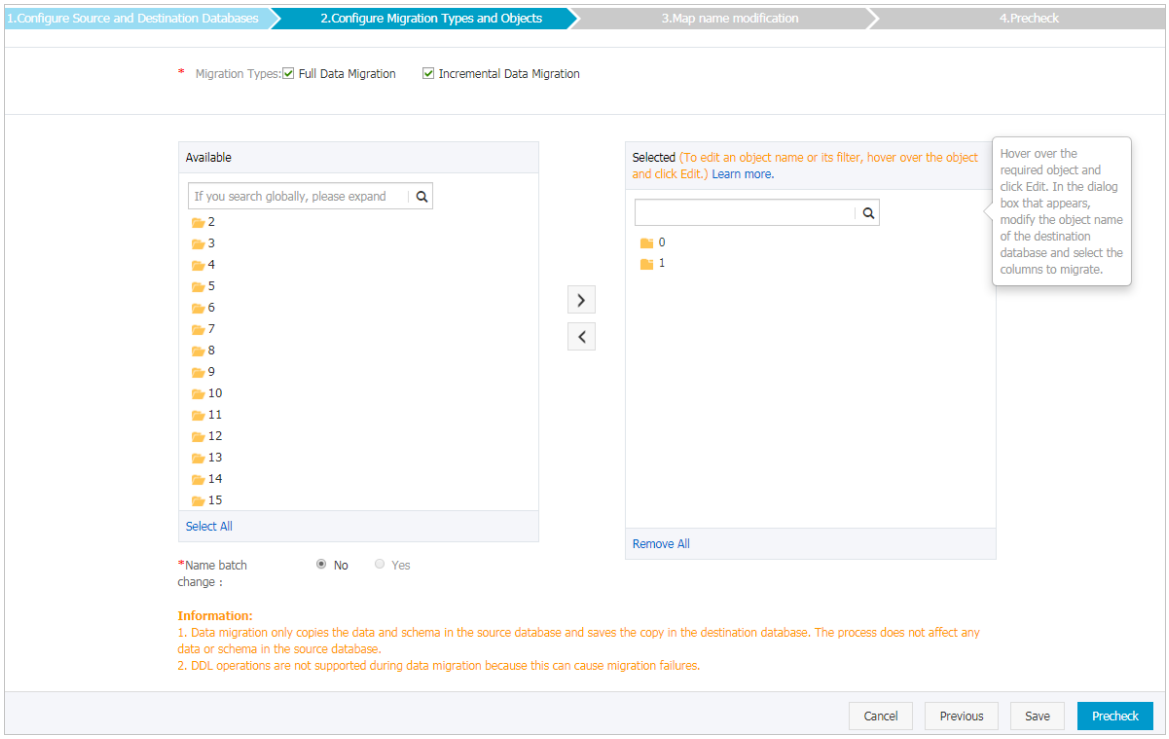
Section	Parameter	Description
N/A	Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
Source Database	Instance Type	Select an instance type based on where the source database is deployed. The procedure in this topic uses **User-Created Database with Public IP Address** as an example. **Note** If you select other instance types, you must prepare the environments that are required for the source database. For more information, see Preparation overview.
	Instance Region	If the instance type is set to **User-Created Database with Public IP Address**, you do not need to specify the **instance region**. **Note** If a whitelist is configured for the user-created Redis database, you must add the CIDR blocks of DTS servers to the whitelist of the user-created Redis database. You can click **Get IP Address Segment of DTS** next to **Instance Region** to obtain the CIDR blocks of DTS servers.
	Database Type	Select **Redis**.

Section	Parameter	Description
	Instance Mode	The value of this parameter is set to **Standalone** and cannot be changed to Cluster.
	Hostname or IP Address	Enter the endpoint that is used to connect to the user-created Redis database. In this example, enter the public IP address.
	Port Number	Enter the service port number of the user-created Redis database. The default port number is 6379.
	Database Password	Enter the password for the user-created Redis database. **Note** After you specify the source database parameters, click **Test Connectivity** next to **Database Password** to verify whether the specified parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the source database parameters based on the check results.
Destination Database	Instance Type	Select **Redis Instance**.
	Instance Region	Select the region where the destination ApsaraDB for Redis instance resides.
	Redis Instance ID	Select the ID of the destination ApsaraDB for Redis instance.
	Database Password	Enter the database password of the destination ApsaraDB for Redis instance. **Note** After you specify the destination database parameters, click **Test Connectivity** next to **Database Password** to verify whether the parameters are valid. If the specified parameters are valid, the **Passed** message appears. If the **Failed** message appears, click **Check** next to **Failed**. Modify the destination database parameters based on the check results.

6. In the lower-right corner of the page, click **Set Whitelist and Next**.

   Note The CIDR blocks of DTS servers are automatically added to the whitelist of the destination ApsaraDB for Redis instance. This ensures that DTS servers can connect to the destination ApsaraDB for Redis instance.

7. Select the migration types and objects to be migrated.



Parameter	Description
Migration Types	○ To perform only full data migration, select **Full Data Migration**. ○ To migrate data with minimal downtime, select both **Full Data Migration** and **Incremental Data Migration**. **Note** If **Incremental Data Migration** is not selected, do not write data into the user-created Redis database during full data migration. This ensures data consistency between the source and destination databases.
Objects	Select objects from the **Available** section and click the › icon to move the objects to the **Selected** section. **Note** You can select databases as the objects to be migrated.

8. In the lower-right corner of the page, click **Precheck**.

> **Note**
>
> - Before you can start the data migration task, a precheck is performed. You can start the data migration task only after the task passes the precheck.
> - If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run the precheck again.

9. After the task passes the precheck, click **Next**.
10. In the **Confirm Settings** dialog box, specify the **Channel Specification** and select the **Data Transmission Service (Pay-As-You-Go) Service Terms**.
11. Click **Buy and Start** to start the migration task.
    - Full data migration

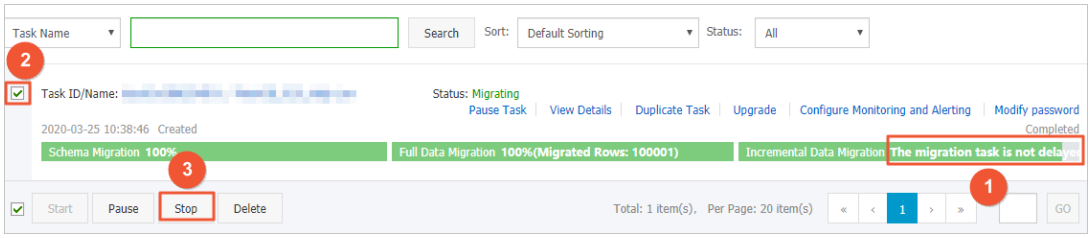
      Do not manually stop a task during full data migration. Otherwise, data migrated to the destination database will be incomplete. Wait until the migration task automatically stops.
    - Incremental data migration

      An incremental data migration task does not automatically stop. You must manually stop the migration task.

      > **Note** Select an appropriate time to manually stop the migration task. For example, you can stop the migration task during off-peak hours or before you switch your workloads to the destination instance.

      a. Wait until **Incremental Data Migration** and **The migration task is not delayed** appear in the progress bar of the migration task. Then, stop writing data to the source database for a few minutes. The delay time of **incremental data migration** may be displayed in the progress bar.

      b. After the status of **incremental data migration** changes to **The migration task is not delayed**, manually stop the migration task.



12. Switch your workloads to the destination ApsaraDB for Redis instance.

What to do next

The database accounts used for data migration have the read/write permissions. After the data migration is complete, you must delete the accounts of both the user-created Redis database and the ApsaraDB for Redis instance to ensure database security.

1.2.2. Use redis-shake to migrate data

You can use the sync mode of redis-shake to migrate data from on-premises Redis to ApsaraDB for Redis.

> Note To migrate the data of an on-premises Codis or Redis cluster to the cloud, see Use the redis-shake tool to migrate the data of an on-premises Codis or Redis cluster to the cloud.

Prerequisites

- An ApsaraDB for Redis instance is created as the destination of data migration.
- An Elastic Compute Service (ECS) instance installed with the Linux operating system is created for running redis-shake.
- The IP address of the ECS instance is added to the whitelist of the destination ApsaraDB for Redis instance.

Background

Redis-shake is an open-source tool developed by Alibaba Cloud. You can use it to parse (decode mode), recover (restore mode), back up (dump mode), and synchronize (sync/rump mode) Redis data. In sync mode, redis-shake runs a **SYNC** or **PSYNC** command to synchronize full or incremental data from the source Redis to the destination Redis. Incremental synchronization automatically starts after full synchronization is completed. The sync mode applies to scenarios such as data migration from on-premises Redis to the cloud, data synchronization between on-premises Redis and ApsaraDB for Redis, and data synchronization between on-premises Redis databases. This topic describes how to use the sync mode of redis-shake to migrate data from on-premises Redis to ApsaraDB for Redis.

> Note
> - In sync mode, the source Redis must support the **SYNC** and **PSYNC** commands. If you use the sync mode of redis-shake to synchronize data from ApsaraDB for Redis, you need to use a database account that has the Replicate permission to connect to the source ApsaraDB for Redis instance.
> - The sync mode supports data synchronization between different Redis versions, such as between Redis 2.8 and Redis 4.0.
> - Currently, ApsaraDB for Redis instances of the cluster edition cannot act as the source of data migration in sync mode.
> - For more information about redis-shake, see redis-shake on GitHub or FAQ.

Procedure

1. Log on to the ECS instance that can access the destination ApsaraDB for Redis instance.
2. Download the redis-shake tool in the ECS instance.

   > Note We recommend that you download the latest version.

3. Run the following command to decompress the downloaded *redis-shake.tar.gz* package:

   ```
   # tar -xvf redis-shake.tar.gz
   ```

> **Note** In the decompressed folder, the *redis-shake* file is a binary file that can be run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of redis-shake. You need to modify this configuration file in the next step.

4. Modify the redis-shake.conf file. The following table describes the parameters for the sync mode of redis-shake.

Parameters for the sync mode of redis-shake

Parameter	Description	Example
source.address	The connection address and service port of the source on-premises Redis instance.	`xxx.xxx. 1.10:6379`
source.password_raw	The password of the source on-premises Redis instance.	`SourcePass233` **Note** If you synchronize data from ApsaraDB for Redis, you need to use a database account that has the Replicate permission to connect to the source ApsaraDB for Redis instance. In this case, set this parameter in the following format: `account:password`.
target.address	The connection address and service port of the destination ApsaraDB for Redis instance.	r-bp1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379
target.password_raw	The password of the destination ApsaraDB for Redis instance.	`TargetPass233`

Parameter	Description	Example
rewrite	Specifies whether to overwrite the existing keys in the destination ApsaraDB for Redis instance that are identical to those in the RDB file. Valid values: ○ true ○ false Note Default value: true. We recommend that you back up the valid data of the destination ApsaraDB for Redis instance before migration. If you set this parameter to false and any keys are duplicate in the source and destination databases, an error message is returned.	true
target.db	The database to which the data is migrated in the destination ApsaraDB for Redis instance. For example, to migrate data from the source on-premises Redis instance to DB10 of the destination ApsaraDB for Redis instance, set this parameter to 10. If you set this parameter to a value less than 0, data is migrated to DB0.	0
parallel	The number of concurrent threads used to synchronize the RDB file. More concurrent threads improve synchronization performance. Note ○ Minimum value: 1. ○ Maximum value: depends on the server performance. ○ Recommended value: 64.	64

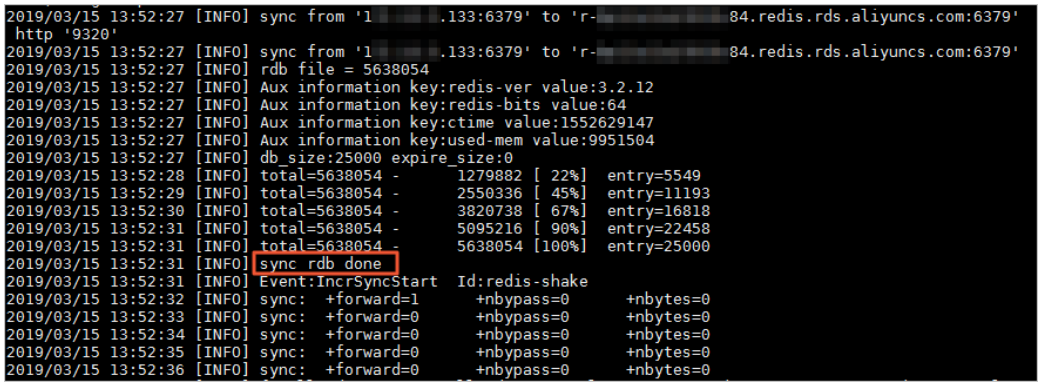
5. Run the following command to migrate data:

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

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you need to specify the correct file path in the command.

6. View synchronization logs to check the synchronization status. When `sync rdb done` appears, full synchronization is completed and incremental synchronization starts.

   Synchronization logs



> **Note**
>
> - In log entries after `sync rdb done`, `+forward=0` indicates that no data is written to the source database and no incremental data is being synchronized. You can monitor the status of data synchronization to determine the best time to switch databases.
> - After migration is completed, you can use the redis-full-check tool to check whether the data is consistent between the source and destination databases. For more information, see Verify data after migration.

1.2.3. Use the redis-shake tool to migrate data from an RDB file

You can use the restore mode of the redis-shake tool to migrate the backup data of an on-premises Redis instance to an ApsaraDB for Redis instance. In this way, you can migrate data from on-premises Redis to the cloud.

Prerequisites

- An ApsaraDB for Redis instance is created as the destination of data migration.
- An Elastic Compute Service (ECS) instance is created for running the redis-shake tool.
- The ECS instance can access the destination ApsaraDB for Redis instance.
- The Linux operating system is running on the ECS instance.
- An RDB file is stored on the ECS instance.

Introduction to the redis-shake tool

The redis-shake tool is an open-source tool developed by Alibaba Cloud. You can use it to parse (decode mode), recover (restore mode), back up (dump mode), and synchronize (sync or rump mode) Redis data. In restore mode, the redis-shake tool can use an RDB file to recover or migrate data. This topic describes how to recover data from an RDB file to an ApsaraDB for Redis instance to help you migrate data from on-premises Redis to the cloud.

> **Note**
> - For more information about the redis-shake tool, see redis-shake on GitHub or FAQ.

Procedure

1. Log on to the ECS instance that can access the destination ApsaraDB for Redis instance.
2. Download the redis-shake tool on the ECS instance.

   > **Note** We recommend that you download the latest version.

3. Run the following command to decompress the *redis-shake.tar.gz* package:

   ```
   # tar -xvf redis-shake.tar.gz
   ```

   > **Note** In the decompressed folder, the *redis-shake* file is a binary file that can run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of the redis-shake tool. You must modify this configuration file in the next step.

4. Modify the *redis-shake.conf* file. The following table describes the parameters for the restore mode of the redis-shake tool.

Parameters for the restore mode of the redis-shake tool

Parameter	Description	Example
rdb.input	The path of the RDB file. You can specify either a relative path or an absolute path.	*/root/tools/RedisShake/demo.rdb*
target.address	The endpoint and port of the destination ApsaraDB for Redis instance.	`r-bp1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379`

Parameter	Description	Example
target.password_raw	The password of the destination ApsaraDB for Redis instance.	TargetPass233 Note If you use a database account other than the default database account to connect to the ApsaraDB for Redis instance, set this parameter in the following format: account:password.
target.db	The database to which the data is recovered in the destination ApsaraDB for Redis instance. Default value: 0. For example, to recover the data of the source on-premises Redis instance to DB10 of the destination ApsaraDB for Redis instance, set this parameter to 10. If you set this parameter to a value less than 0, data is recovered to DB0.	0
rewrite	Specifies whether to overwrite the existing keys that are identical to those in the RDB file. Valid values: ○ true ○ false Note Default value: true. We recommend that you back up the valid data of the destination ApsaraDB for Redis instance before migration. If you set this parameter to false and any keys are duplicate in the source and destination databases, an error message is returned.	true

Parameter	Description	Example
parallel	The number of concurrent threads used to synchronize the RDB file. More concurrent threads improve synchronization performance. **Note** ○ Minimum value: 1. ○ Maximum value: depends on the server performance. ○ Recommended value: 64.	64

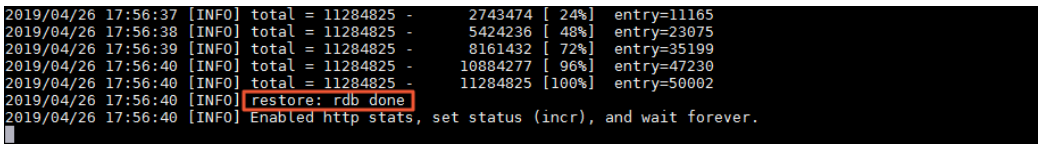
Note You can use the default values for other parameters unless otherwise specified.

5. Run the following command to recover data:

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

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you must specify the correct file path in the command.

Example



Note When `restore: rdb done` appears in logs, the data is recovered. You can press Ctrl+C to exit the tool.

1.2.4. Use the redis-shake tool to migrate the data of an on-premises Codis or Redis cluster to the cloud

You can use the sync mode of the redis-shake tool to migrate the data of an on-premises Codis or Redis cluster to an ApsaraDB for Redis instance.

Note For more information about how to migrate the data of a standalone Redis instance, see Use redis-shake to migrate data.

Prerequisites

- An ApsaraDB for Redis instance is created as the destination of data migration. For more

information about how to create an ApsaraDB for Redis instance, see Step 1: Create an ApsaraDB for Redis instance.

- An Elastic Compute Service (ECS) instance is created for running the redis-shake tool. The 64-bit Linux operating system is running on the ECS instance. For more information about how to create an ECS instance, see Create an ECS instance.
- The ECS instance can access the destination ApsaraDB for Redis instance.

> Note
>
> - If the ECS instance and the destination ApsaraDB for Redis instance are in the same Virtual Private Cloud (VPC), you must add the internal IP address of the ECS instance to the whitelist of the destination ApsaraDB for Redis instance. For more information, see Set IP address whitelists.
> - If the ECS instance and the destination ApsaraDB for Redis instance are not in the same VPC, the ECS instance can access the destination ApsaraDB for Redis instance through the public endpoint. For more information, see Through the Internet.

Limits

- The sync mode applies to scenarios such as data migration from on-premises Redis to the cloud, data synchronization between on-premises Redis and ApsaraDB for Redis, and data synchronization between on-premises Redis databases. Currently, ApsaraDB for Redis instances of the cluster edition cannot act as the source of data migration in sync mode.
- In sync mode, the source Redis must support the **SYNC** and **PSYNC** command.

Introduction to the redis-shake tool

The redis-shake tool is an open-source tool developed by Alibaba Cloud. You can use it to parse (decode mode), recover (restore mode), back up (dump mode), and synchronize (sync or rump mode) Redis data. In sync mode, the redis-shake tool runs the **SYNC** or **PSYNC** command to synchronize full or incremental data from the source Redis to the destination Redis. Incremental synchronization automatically starts after full synchronization is completed. This topic describes how to use the sync mode of the redis-shake tool to migrate the data of an on-premises Codis or Redis cluster to an ApsaraDB for Redis instance.

> Note
>
> - The sync mode supports data synchronization between different Redis versions, for example, between Redis 2.8 and Redis 4.0.
> - For more information about the redis-shake tool, see redis-shake on GitHub or FAQ.

Procedure

1. Log on to the ECS instance that can access the destination ApsaraDB for Redis instance.
2. Download the redis-shake tool on the ECS instance.

   > Note We recommend that you download the latest version.

3. Run the following command to decompress the *redis-shake.tar.gz* package:

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

Note In the decompressed folder, the *redis-shake* file is a binary file that can run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of the redis-shake tool. You must modify this configuration file in the next step.

4. Modify the *redis-shake.conf* file. The following table describes the parameters for the sync mode of the redis-shake tool.

Parameter	Description	Example
source.type	The type of the source Codis or Redis cluster.	`cluster`
source.address	The endpoint and port of the source Codis or Redis cluster.	`10.xx.xx.1:7000;10.xx.xx.1:7002;10.xx.xx.1:7003;10.xx.xx.1:7004`
source.password_raw	The password of the source Codis or Redis cluster.	`SourcePass233`
target.type	The type of the destination ApsaraDB for Redis instance.	`proxy`
target.address	The endpoint and port of the destination ApsaraDB for Redis instance. For more information, see View endpoints.	`r-bpxxxxxxxxxxxxxxx.redis.rds.aliyuncs.com:6379`
target.password_raw	The password of the destination ApsaraDB for Redis instance.	`TargetPass233`
rewrite	Specifies whether to overwrite the existing keys that are identical to those in the RDB file. Valid values: ◦ true ◦ false Note Default value: true. We recommend that you back up the valid data of the destination ApsaraDB for Redis instance before migration. If you set this parameter to false and any keys are duplicate in the source and destination databases, an error message is returned.	`true`

Parameter	Description	Example
target.db	The database to which the data is migrated in the destination ApsaraDB for Redis instance. For example, to migrate data from the source on-premises Codis or Redis cluster to DB10 of the destination ApsaraDB for Redis instance, set this parameter to 10. If you set this parameter to -1, data in the source Codis or Redis cluster is migrated to the same database in the destination ApsaraDB for Redis instance. For example, the data of DB0 in the source Codis or Redis cluster is migrated to DB0 in the destination ApsaraDB for Redis instance, the data of DB1 in the source Codis or Redis cluster is migrated to DB1 in the destination ApsaraDB for Redis instance, and so on.	0
parallel	The number of concurrent threads used to synchronize the RDB file. More concurrent threads improve synchronization performance. Note ◦ Minimum value: 1. ◦ Maximum value: depends on the server performance. ◦ Recommended value: 64.	64

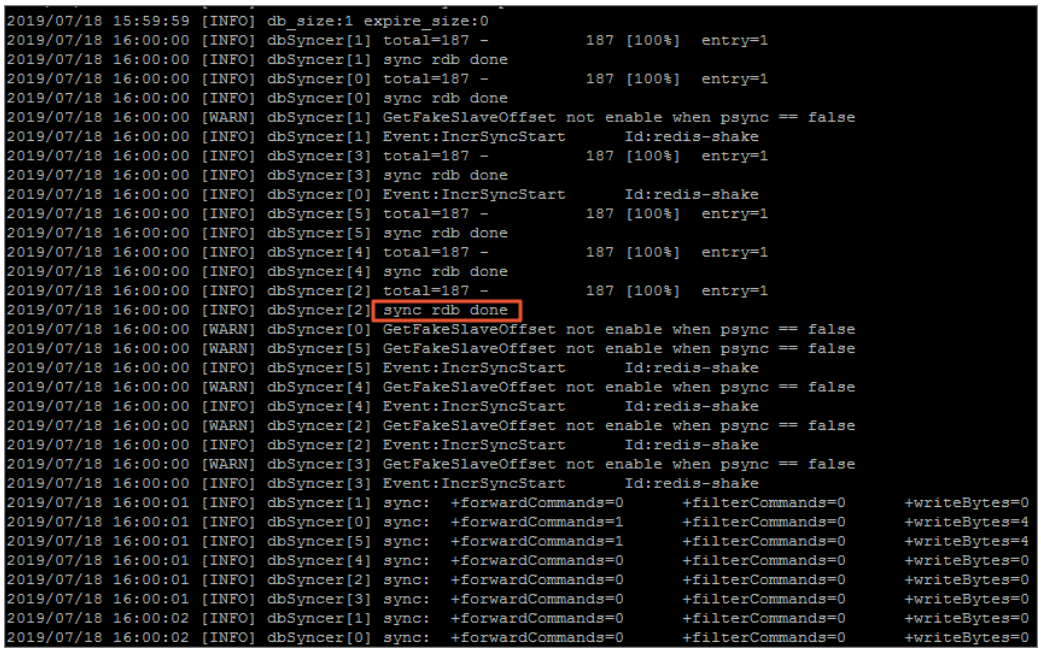
5. Run the following command to migrate data:

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

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you must specify the correct file path in the command.

6. View synchronization logs to check the synchronization status. When `sync rdb done` appears, full synchronization is completed and incremental synchronization starts.

Synchronization logs



> **Note** Whenever the data of a node in the Codis or Redis cluster is synchronized, `sync rdb done` appears.

7. After the data of all nodes is synchronized, `+forwardCommands=0` indicates that no data is written to the source database and no incremental data is being synchronized. In this case, switch your services to the new database.

1.2.5. Synchronize data from a Codis cluster hosted on ECS to an ApsaraDB for Redis instance

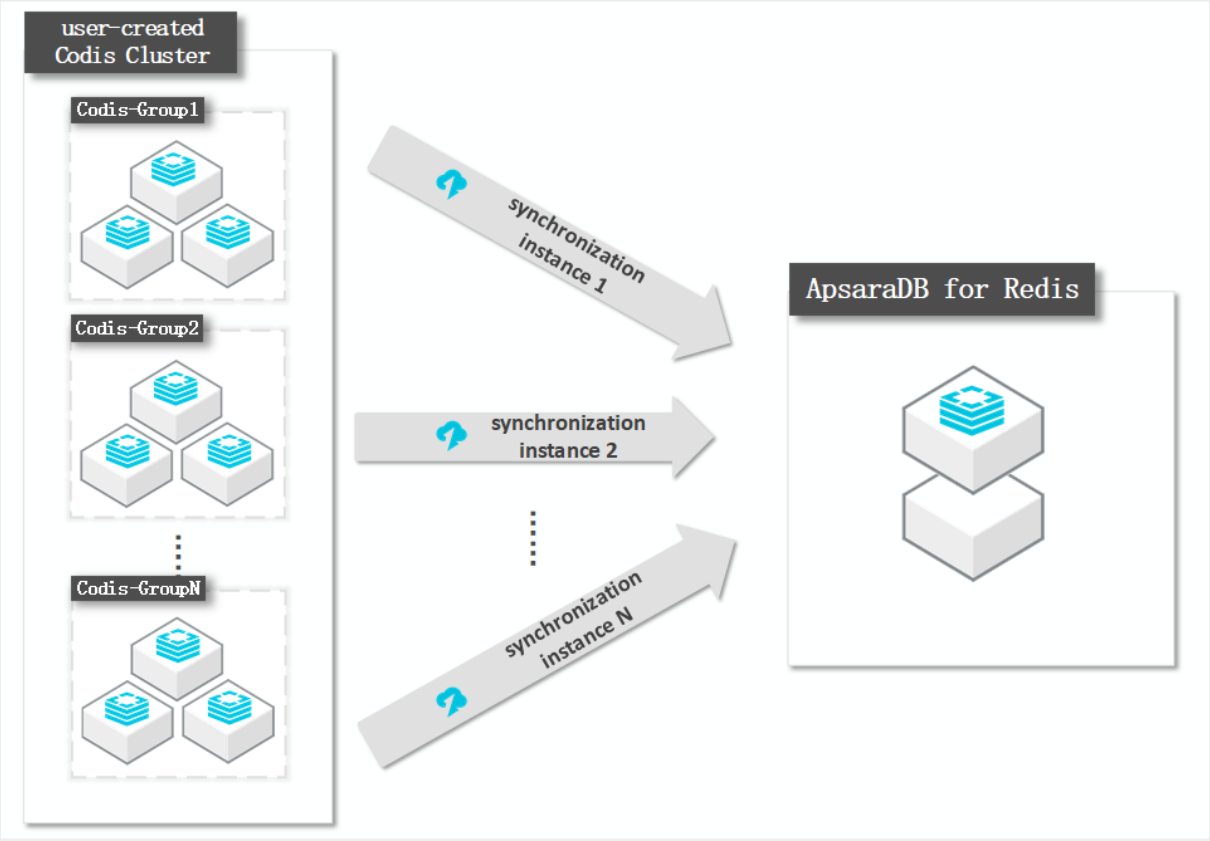
ApsaraDB for Redis is a database service compatible with the open source Redis protocol and provides hybrid storage of memory and hard disks. Based on reliable hot standby architecture and scalable cluster architecture, ApsaraDB for Redis is suitable for scenarios that require flexible configuration changes, high throughput, and low latency. This topic describes how to synchronize data from a Codis cluster to an ApsaraDB for Redis instance by using Data Transmission Service (DTS).

Prerequisites

- A destination ApsaraDB for Redis instance that is used Redis 2.8, 4.0, or 5.0 is created. For more information, see Create an ApsaraDB for Redis instance.
- The available storage space of the destination ApsaraDB for Redis instance is larger than the total size of data stored in the source Codis cluster.
- All master nodes in the source Codis cluster support the `PSYNC` command.

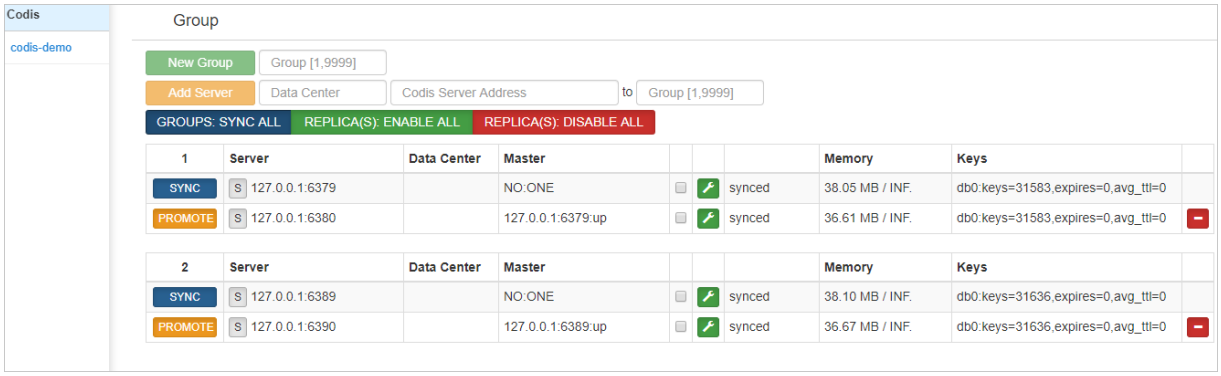
How DTS synchronizes data from a Codis cluster

A Codis cluster consists of multiple Codis-Groups. You must create a data synchronization task for each Codis-Group. DTS synchronizes each Codis-Group in a data synchronization task until the whole cluster is synchronized.



Architecture of the Codis cluster

In this topic, the Codis cluster consists of two Codis-Groups. Each Codis-Group runs in a master-replica architecture. The following figure shows the architecture of the cluster.



Notes

- DTS uses resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. If you synchronize a large volume of data or the server specifications cannot meet your requirements, database services may become unavailable. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.

- If the `bind` parameter is configured in the `redis.conf` file of the source Redis database, set the value of this parameter to the intranet IP address of ECS to ensure that DTS can connect to the source database normally.
- We recommend that you increase the value of the `repl-backlog-size` parameter in the *redis.conf* file. This ensures the stability of data synchronization.
- To ensure the synchronization quality, DTS adds the following key to the source Codis cluster: `DTS_REDIS_TIMESTAMP_HEARTBEAT`. This key is used to record the time when data is synchronized to ApsaraDB for Redis.
- We recommend that you do not run the `FLUSHDB` or `FLUSHALL` command in the source Codis cluster. Otherwise, data may be inconsistent between the Codis cluster and the ApsaraDB for Redis instance after data synchronization.
- If the data eviction policy (`maxmemory-policy`) of the destination instance is not set to `noeviction` , data may become inconsistent between the source and destination instances. For more information about the data eviction policy, see How does ApsaraDB for Redis evict data by default?
- The database version of the destination ApsaraDB for Redis instance must be 2.8, 4.0, or 5.0. The version of the destination database must be the same as or later than the version of the source database. If you want to synchronize data between different versions of Redis databases, make sure that the versions of the source and destination databases are compatible with each other. You can create a pay-as-you-go ApsaraDB for Redis instance to verify database compatibility. After the verification, you can release the instance or change the billing method to subscription.

Supported synchronization topologies

- One-way one-to-one synchronization
- One-way one-to-many synchronization
- One-way cascade synchronization

For more information about synchronization topologies, see Synchronization topologies.

Supported commands for data synchronization

- APPEND
- BITOP, BLPOP, BRPOP, and BRPOPLPUSH
- DECR, DECRBY, and DEL
- EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT
- GEOADD and GETSET
- HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX
- INCR, INCRBY, and INCRBYFLOAT
- LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM
- MOVE, MSET, MSETNX, and MULTI
- PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, PSETEX, and PUBLISH
- RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX
- SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE

- ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

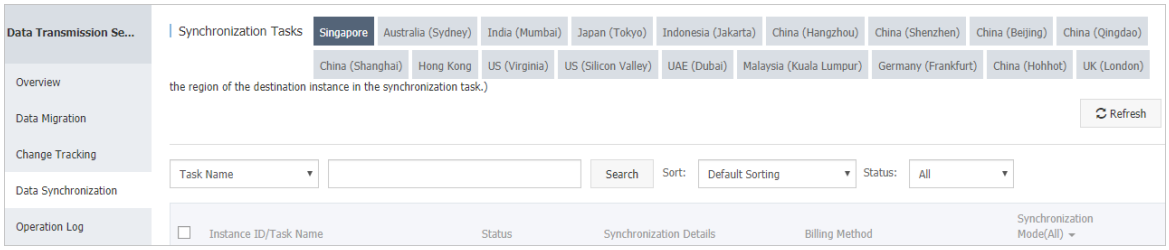
> **Note**
> - If you run the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination database. During incremental data synchronization, the destination database does not explicitly return the execution results of Lua scripts.
> - When DTS calls the SYNC or PSYNC command to transfer data of the LIST type, DTS does not clear the existing data. In this case, the destination database may contain duplicate data records.

Procedure

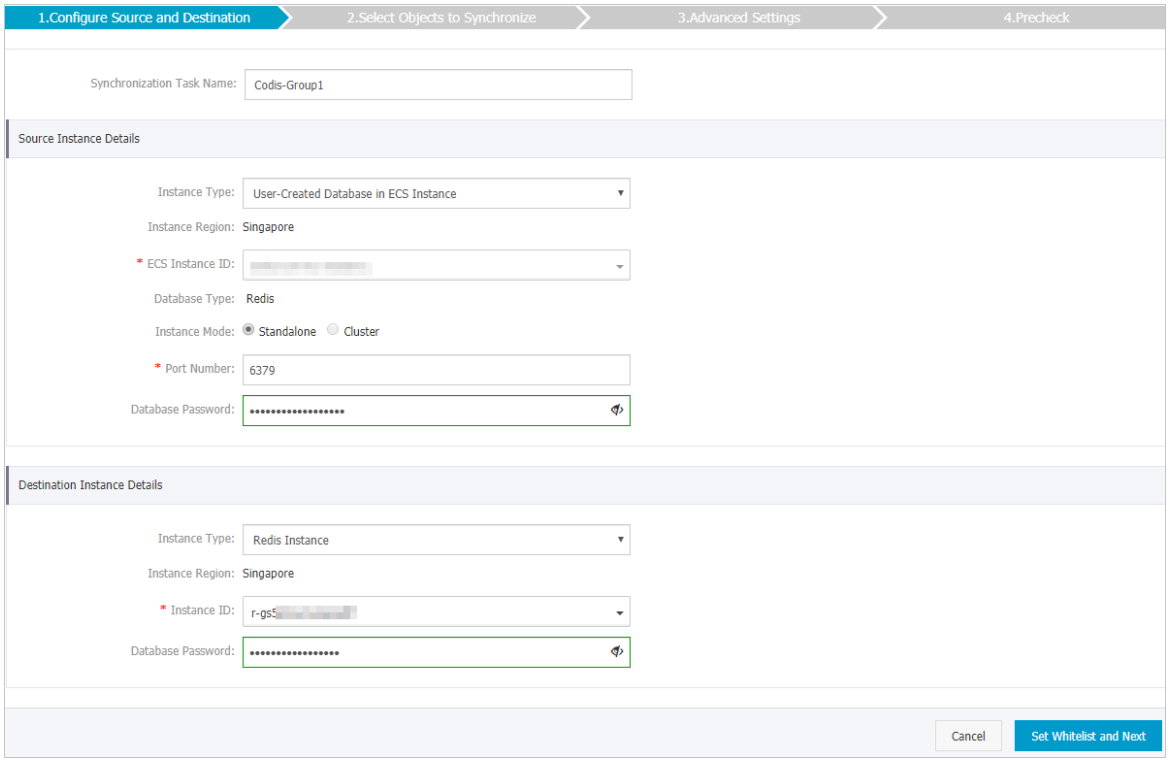
1. Purchase a data synchronization instance. For more information, see Purchase procedure.

   > **Note** On the buy page, set Source Instance to **Redis**, set Target Instance to **Redis**, and then set Synchronization Topology to **One-Way Synchronization**.

2. Log on to the DTS console.
3. In the left-side navigation pane, click **Data Synchronization**.
4. At the top of the **Synchronization Tasks** page, select the region where the destination instance resides.



5. Find the data synchronization instance and click **Configure Synchronization Channel** in the Actions column.
6. Configure the source and destination instances.



Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you specify an informative name for easy identification. You do not need to use a unique task name.
	Instance Type	The type of the source instance. Select **User-Created Database in ECS Instance**.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you purchased the data synchronization instance. You cannot change the value of this parameter.

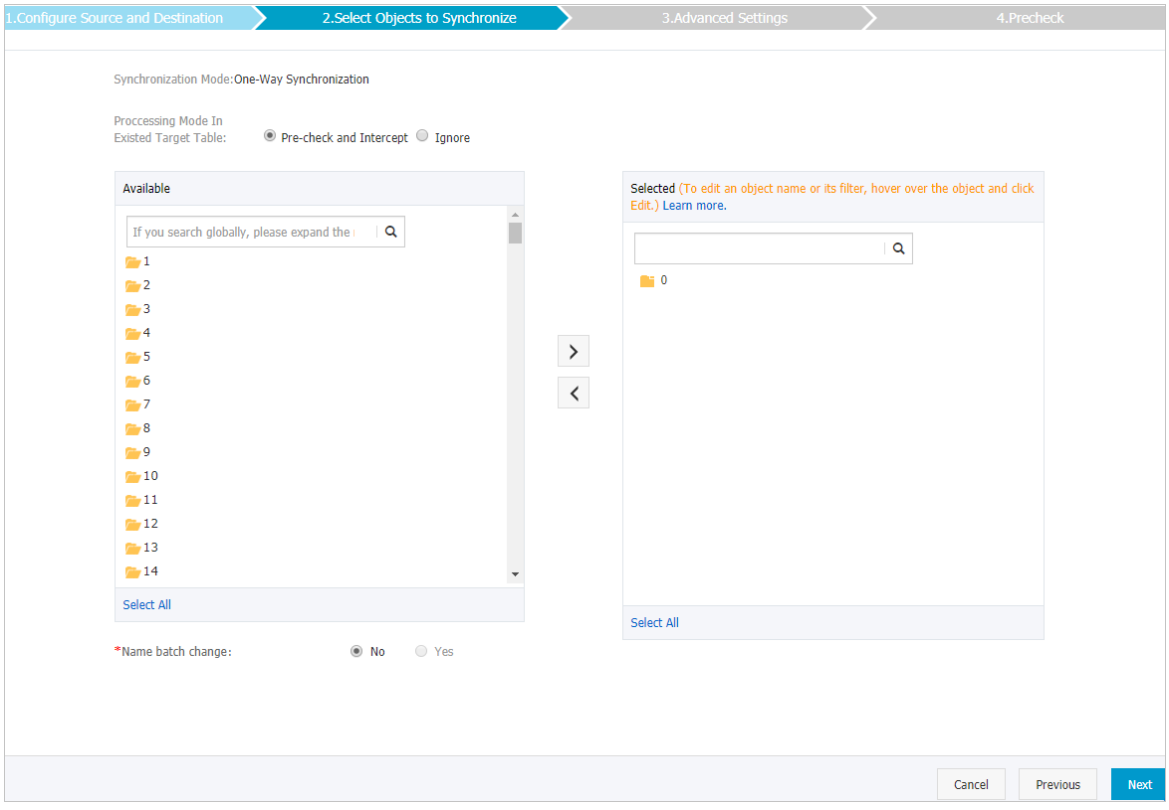
Section	Parameter	Description
Source Instance Details	ECS Instance ID	Select the ID of the ECS instance where the master node of the Codis-Group is deployed. Note DTS synchronizes each Codis-Group of the Codis cluster by using a data synchronization task till the whole cluster is synchronized. In this step, enter the ECS instance ID for the master node of Codis-Group 1. When you configure the data synchronization task for Codis-Group 2, enter the ECS instance ID for the master node of Codis-Group 2. You can configure data synchronization tasks for all Codis-Groups by following the procedure described in this topic.
	Database Type	The value of this parameter is set to **Redis**.
	Instance Mode	Select **Standalone**. Note You must select **Standalone** for this parameter because data from a Codis cluster cannot be synchronized at a time. DTS synchronizes each Codis-Group of the cluster in a data synchronization task till all Codis-Groups are synchronized.
	Port Number	Enter the service port number of the master node in the Codis-Group.
	Database Password	The database password for the master node. Note This parameter is optional and can be left blank if no database password is set.
	Instance Type	Select **Redis Instance**.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you purchased the data synchronization instance. You cannot change the value of this parameter.

Section	Parameter	Description
Destination Instance Details	Instance ID	Select the ID of the destination ApsaraDB for Redis instance.
	Database Password	The database password of the destination ApsaraDB for Redis instance. Note The format of the database password is <user>:<password>. For example, if the username of the custom account is admin and the password is Rp829dlwa, the database password is admin:Rp829dlwa.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

   Note The CIDR blocks of DTS servers are automatically added to the inbound rule of the ECS instance and the whitelist of the ApsaraDB for Redis instance. This ensures that DTS servers can connect to the source and destination instances.

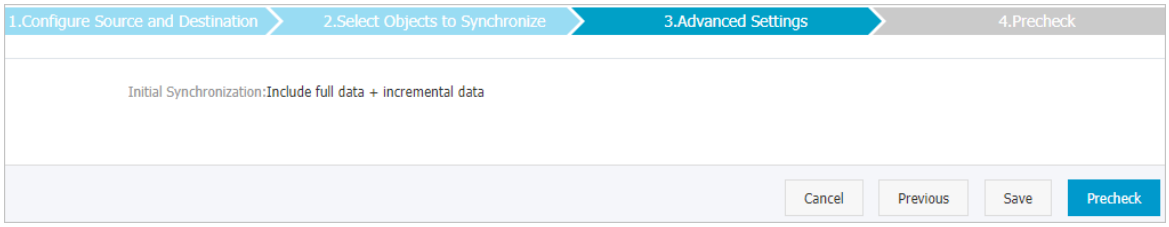
8. Configure the processing mode in existing destination tables and the objects to be synchronized.



Parameter	Description

Parameter	Description
Processing Mode In Existed Target Table	DTS synchronizes each Codis-Group of the Codis cluster in a data synchronization task till the whole cluster is synchronized. When you configure data synchronization for Codis-Group 1, if the ApsaraDB for Redis instance has no data, select **Pre-check and Intercept**. When you configure data synchronization for Codis-Groups 2 to N, select **Ignore**. Otherwise, errors may occur during data synchronization. **Note** ○ **Pre-check and Intercept**: checks whether the destination database is empty. If the destination database is empty, the precheck is passed. If the database is not empty, an exception is returned during precheck and the data synchronization task cannot be started. ○ **Ignore**: skips the precheck for empty destination databases and continues with data synchronization. If the keys in the destination database are the same as those in the source database during data synchronization, the keys in the source database overwrite those in the destination database.
Objects to be synchronized	○ Select one or more databases from the **Available** section and click > to move the databases to the **Selected** section. ○ You can select only databases as the objects to be synchronized. Keys are not supported.

9. In the lower-right corner of the page, click **Next**.
10. Configure initial synchronization.

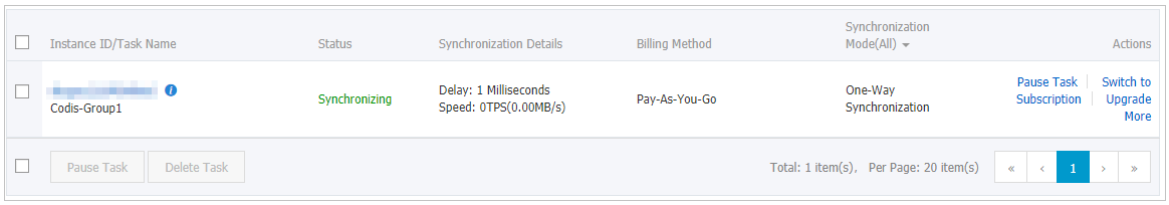


Note The value is set to **Include full data + incremental data**. DTS synchronizes historical data from the source Codis cluster to the destination Redis database, and then synchronizes incremental data.

11. In the lower-right corner of the page, click **Precheck**.

> **Note**
>
> - Before you can start the data synchronization task, a precheck is performed. You can start the data synchronization task only after the task passes the precheck.
> - If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run the precheck again.

12. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
13. Wait until the initial synchronization is complete and the data synchronization task is in the **Synchronizing** state.

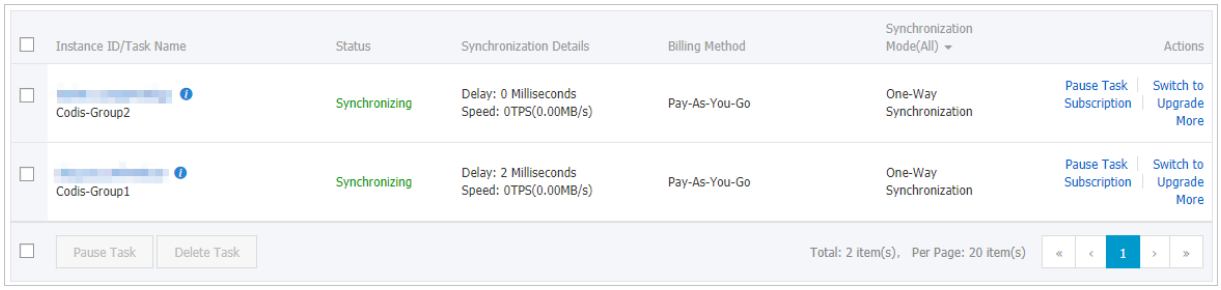


> **Note** You can view the status of the data synchronization task on the **Synchronization Tasks** page.

14. Create and configure a data synchronization task for the other Codis-Group by following Steps 1 to 13.

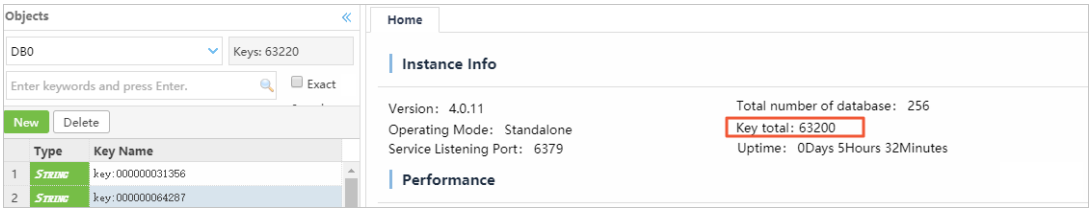
Result

In this topic, the Codis cluster consists of two Codis-Groups. You must create two data synchronization tasks. The following figure shows that the initial synchronization is complete for both tasks and both tasks are in the **Synchronizing** state.

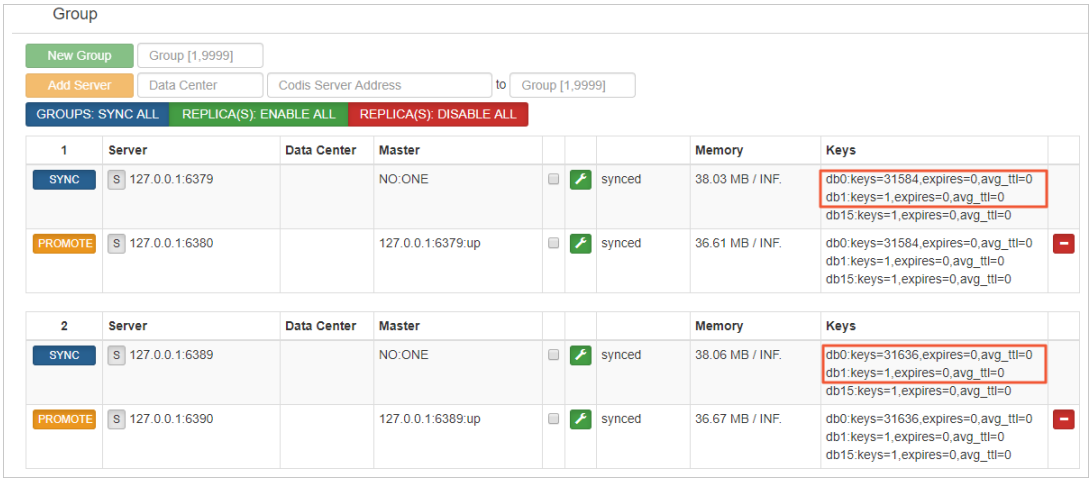


In this topic, databases DB0 and DB1 are synchronized. You can use Data Management (DMS) to log on to the ApsaraDB for Redis instance and check the total number of keys in the ApsaraDB for Redis instance. The total number of keys is the same as that in the source Codis cluster.

ApsaraDB for Redis instance



Source Codis cluster



1.2.6. Synchronize data from a Twemproxy Redis cluster hosted on ECS to an ApsaraDB for Redis instance

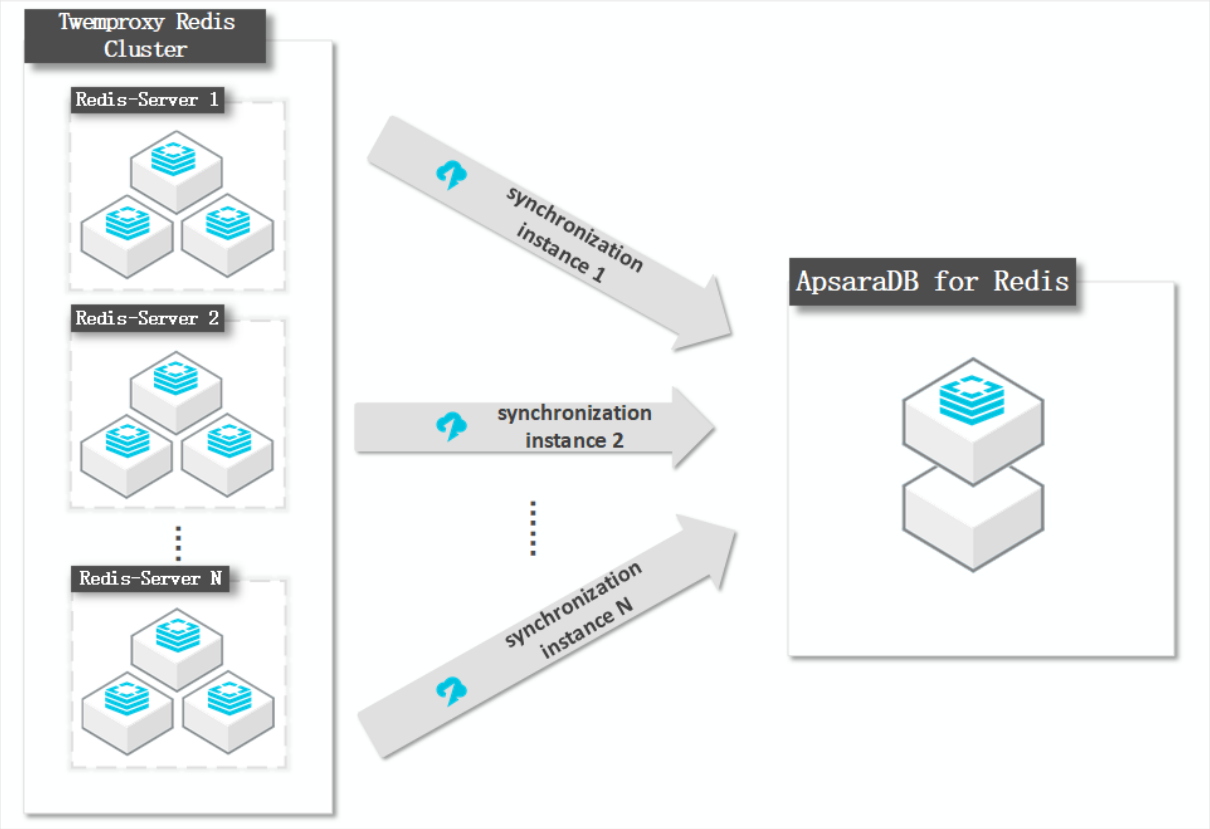
ApsaraDB for Redis is a database service compatible with the open-source Redis protocol and provides hybrid storage of memory and hard disks. Based on reliable hot standby architecture and scalable cluster architecture, ApsaraDB for Redis is suitable for scenarios that require flexible configuration changes, high throughput, and low latency. This topic describes how to synchronize data from a Twemproxy Redis cluster to an ApsaraDB for Redis instance by using Data Transmission Service (DTS).

Prerequisites

- The available storage space of the destination ApsaraDB for Redis instance is larger than the total size of data stored in the source Twemproxy Redis cluster.
- All master nodes in the source Twemproxy Redis cluster support the `PSYNC` command.

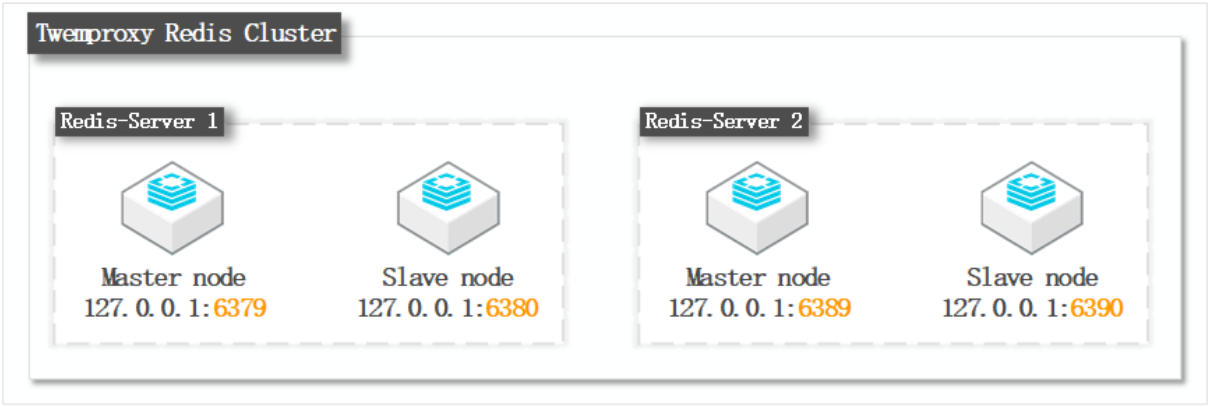
How DTS synchronizes data from a Twemproxy Redis cluster

A Twemproxy Redis cluster consists of multiple Redis-Servers. DTS synchronizes each Redis-Server in a data synchronization task till the whole cluster is synchronized.



Architecture of the Twemproxy Redis cluster

In this topic, the Twemproxy Redis cluster consists of two Redis-Servers. Each Redis-Server runs in a master-replica architecture. The following figure shows the architecture of the cluster.



Notes

- DTS uses resources of the source and destination databases during initial full data synchronization. This may increase the load of the database server. If you synchronize a large volume of data or the server specifications cannot meet your requirements, database services may become unavailable. Before you synchronize data, evaluate the impact of data synchronization on the performance of the source and destination databases. We recommend that you synchronize data during off-peak hours.
- If the `bind` parameter is configured in the `redis.conf` file of the source Redis database, set the value of this parameter to the intranet IP address of ECS to ensure that DTS can connect

to the source database normally.

- We recommend that you increase the value of the `repl-backlog-size` parameter in the *redis.conf* file. This ensures the stability of data synchronization.
- To ensure the synchronization quality, DTS adds the following key to the source Codis cluster: `DTS_REDIS_TIMESTAMP_HEARTBEAT`. This key is used to record the time when data is synchronized to ApsaraDB for Redis.
- We recommend that you do not run the `FLUSHDB` or `FLUSHALL` command in the source Codis cluster. Otherwise, data may be inconsistent between the Codis cluster and the ApsaraDB for Redis instance after data synchronization.
- If the data eviction policy (`maxmemory-policy`) of the destination instance is not set to `noeviction` , data may become inconsistent between the source and destination instances. For more information about the data eviction policy, see How does ApsaraDB for Redis evict data by default?
- The database version of the destination ApsaraDB for Redis instance must be 2.8, 4.0, or 5.0. The version of the destination database must be the same as or later than the version of the source database. If you want to synchronize data between different versions of Redis databases, make sure that the versions of the source and destination databases are compatible with each other. You can create a pay-as-you-go ApsaraDB for Redis instance to verify database compatibility. After the verification, you can release the instance or change the billing method to subscription.

Supported synchronization topologies

- One-way one-to-one synchronization
- One-way one-to-many synchronization
- One-way cascade synchronization

For more information about synchronization topologies, see Synchronization topologies.

Supported commands for data synchronization

- APPEND
- BITOP, BLPOP, BRPOP, and BRPOPLPUSH
- DECR, DECRBY, and DEL
- EVAL, EVALSHA, EXEC, EXPIRE, and EXPIREAT
- GEOADD and GETSET
- HDEL, HINCRBY, HINCRBYFLOAT, HMSET, HSET, and HSETNX
- INCR, INCRBY, and INCRBYFLOAT
- LINSERT, LPOP, LPUSH, LPUSHX, LREM, LSET, and LTRIM
- MOVE, MSET, MSETNX, and MULTI
- PERSIST, PEXPIRE, PEXPIREAT, PFADD, PFMERGE, PSETEX, and PUBLISH
- RENAME, RENAMENX, RESTORE, RPOP, RPOPLPUSH, RPUSH, and RPUSHX
- SADD, SDIFFSTORE, SELECT, SET, SETBIT, SETEX, SETNX, SETRANGE, SINTERSTORE, SMOVE, SPOP, SREM, and SUNIONSTORE
- ZADD, ZINCRBY, ZINTERSTORE, ZREM, ZREMRANGEBYLEX, ZUNIONSTORE, ZREMRANGEBYRANK, and ZREMRANGEBYSCORE

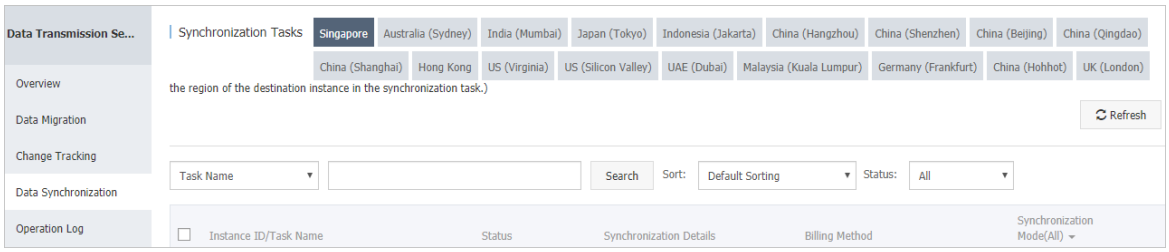
> **Note**
>
> - If you run the EVAL or EVALSHA command to call Lua scripts, DTS cannot identify whether these Lua scripts are executed on the destination database. During incremental data synchronization, the destination database does not explicitly return the execution results of Lua scripts.
> - When DTS calls the SYNC or PSYNC command to transfer data of the LIST type, DTS does not clear the existing data. In this case, the destination database may contain duplicate data records.

Procedure

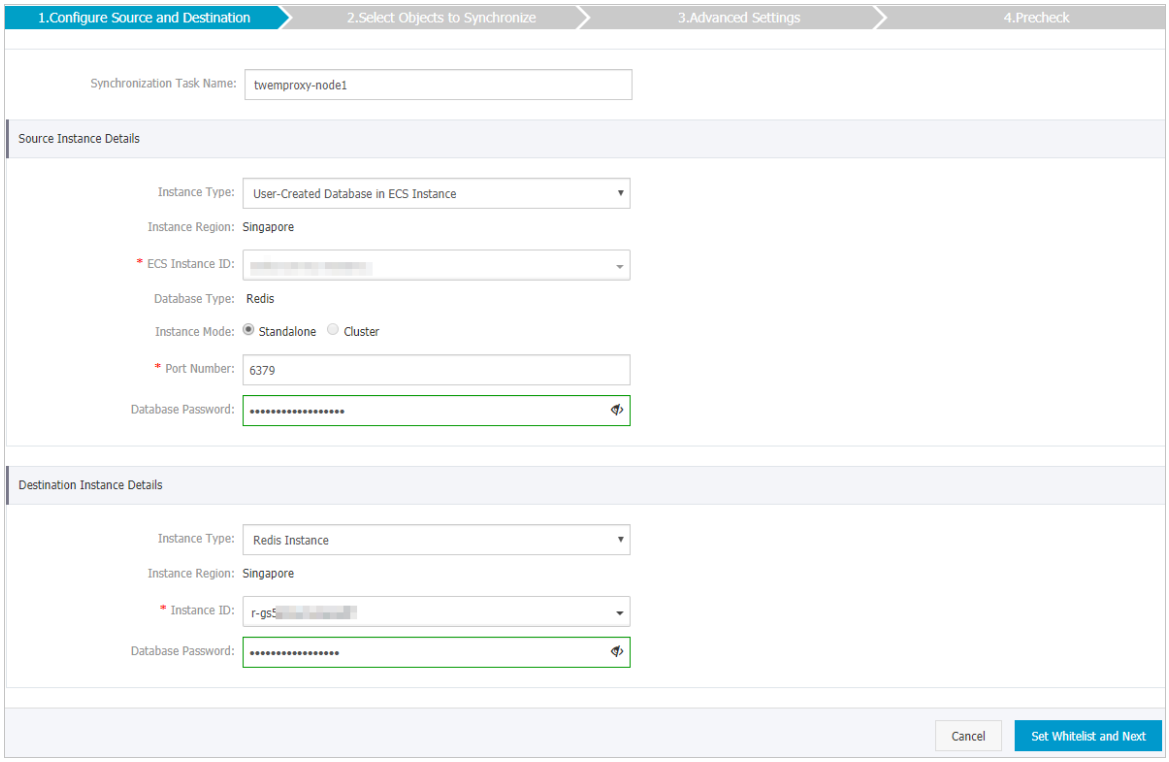
1. Purchase a data synchronization instance. For more information, see Purchase procedure.

   > **Note** On the buy page, set Source Instance to **Redis**, Target Instance to **Redis**, and Synchronization Topology to **One-Way Synchronization**.

2. Log on to the DTS console.
3. In the left-side navigation pane, click **Data Synchronization**.
4. At the top of the **Synchronization Tasks** page, select the region where the destination instance resides.



5. Find the data synchronization instance and click **Configure Synchronization Channel** in the Actions column.
6. Configure the source and destination instances.



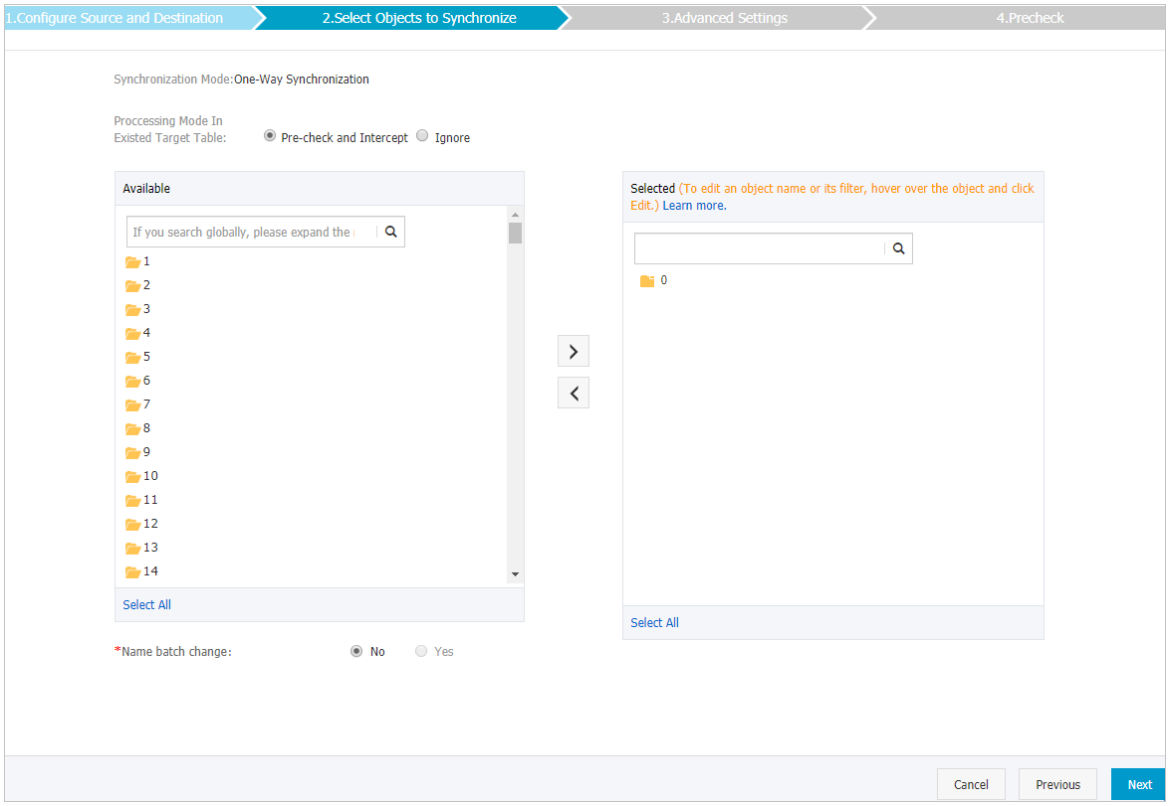
Section	Parameter	Description
N/A	Synchronization Task Name	DTS automatically generates a task name. We recommend that you use an informative name for easy identification. You do not need to use a unique task name.
Source Instance Details	Instance Type	Select **User-Created Database in ECS Instance**.
	Instance Region	The region of the source instance. The region is the same as the source region that you selected when you purchased the data synchronization instance. You cannot change the value of this parameter.
	ECS Instance ID	Select the ID of the ECS instance where the master node of the Redis-Server resides. **Note** DTS synchronizes each Redis-Server of the Twemproxy Redis cluster by using a data synchronization task till the whole cluster is synchronized. In this step, enter the ECS instance ID for the master node of Redis-Server 1. When you configure the data synchronization task for Redis-Server 2, enter the ECS instance ID for the master node of Redis-Server 2. You can configure data synchronization tasks for all Redis-Servers by following the procedure described in this topic.
	Database Type	The value of this parameter is set to **Redis**.

Section	Parameter	Description
	Instance Mode	Select **Standalone**. **Note** You must select **Standalone** for this parameter because data from a Twemproxy Redis cluster cannot be synchronized at a time. DTS synchronizes each Redis-Server of the cluster in a data synchronization task till all Redis-Servers are synchronized.
	Port Number	Enter the service port number of the master node in the Redis-Server.
	Database Password	Enter the database password for the master node. **Note** This parameter is optional and can be left blank if no database password is set.
Destination Instance Details	Instance Type	Select **Redis Instance**.
	Instance Region	The region of the destination instance. The region is the same as the destination region that you selected when you purchased the data synchronization instance. You cannot change the value of this parameter.
	Instance ID	Select the ID of the destination ApsaraDB for Redis instance.
	Database Password	Enter the database password of the destination ApsaraDB for Redis instance. **Note** The format of the database password is <user>:<password>. For example, if the username of the custom account is admin and the password is Rp829dlwa, the database password is admin:Rp829dlwa.

7. In the lower-right corner of the page, click **Set Whitelist and Next**.

   Note The CIDR blocks of DTS servers are automatically added to the inbound rule of the ECS instance and the whitelist of the ApsaraDB for Redis instance. This ensures that DTS servers can connect to the source and destination instances.

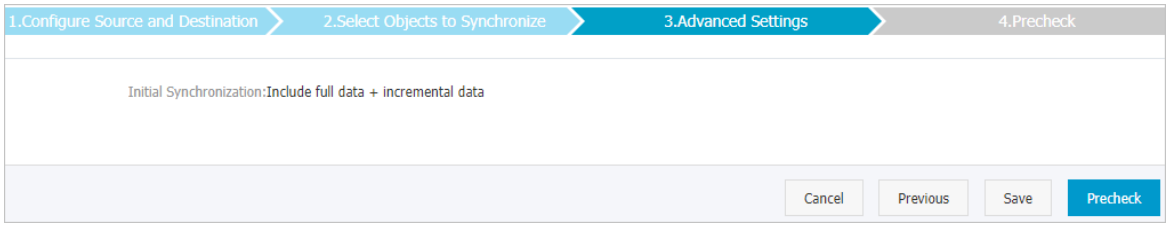
8. Configure the processing mode in existing destination tables and the objects to be synchronized.



Parameter	Description
Processing Mode In Existed Target Table	DTS synchronizes each Redis-Server of the Twemproxy Redis cluster in a data synchronization task till the whole cluster is synchronized. When you configure data synchronization for Redis-Server 1, if the ApsaraDB for Redis instance has no data, select **Pre-check and Intercept**. When you configure data synchronization for Redis-Server 2 to N, select **Ignore**. Otherwise, errors may occur during data synchronization. **Note** ○ **Pre-check and Intercept**: checks whether the destination database is empty. If the destination database is empty, the precheck is passed. If the database is not empty, an exception is returned during precheck and the data synchronization task cannot be started. ○ **Ignore**: skips the precheck for empty destination databases and continues with data synchronization. If the keys in the destination database are the same as those in the source database during data synchronization, the keys in the source database overwrite those in the destination database.

Parameter	Description
Objects to be synchronized	○ Select one or more databases from the **Available** section and click > to move the databases to the **Selected** section. ○ You can select only databases as the objects to be synchronized. Keys are not supported.

9. In the lower-right corner of the page, click **Next**.
10. Configure initial synchronization.

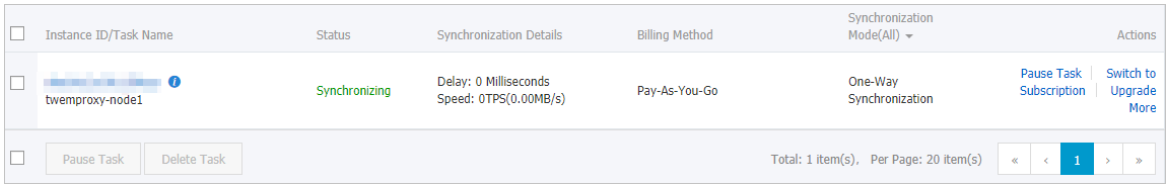


> **Note** The value is set to **Include full data + incremental data**. DTS synchronizes historical data from the source Twemproxy Redis cluster to the destination Redis database before synchronizing incremental data.

11. In the lower-right corner of the page, click **Precheck**.

> **Note**
> ○ Before you can start the data synchronization task, a precheck is performed. You can start the data synchronization task only after the task passes the precheck.
> ○ If the task fails to pass the precheck, click the icon next to each failed item to view details. Troubleshoot the issues based on the causes and run the precheck again.

12. Close the **Precheck** dialog box after the following message is displayed: **The precheck is passed.**
13. Wait until the initial synchronization is complete and the data synchronization task is in the **Synchronizing** state.

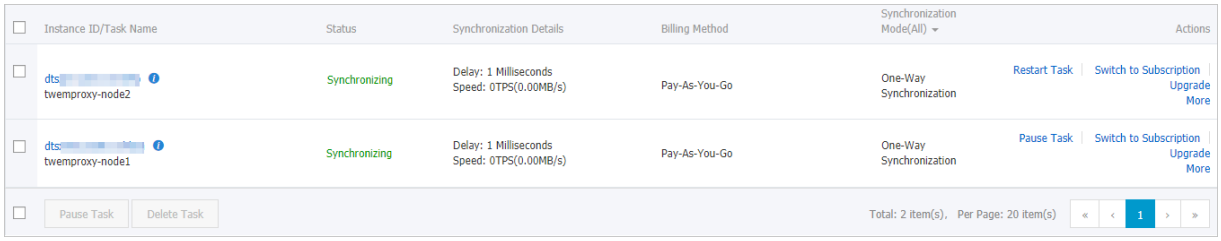


> **Note** You can view the status of the data synchronization task on the **Synchronization Tasks** page.

14. Create and configure a data synchronization task for every other Redis-Server by repeating steps 1 to 13.

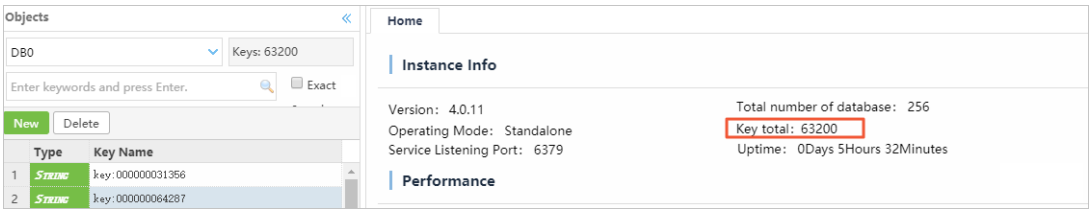
Result

In this topic, the Twemproxy Redis cluster consists of two Redis-Servers. You must create two data synchronization tasks. The following figure shows that the initial synchronization is complete for both tasks and both tasks are in the **Synchronizing** state.

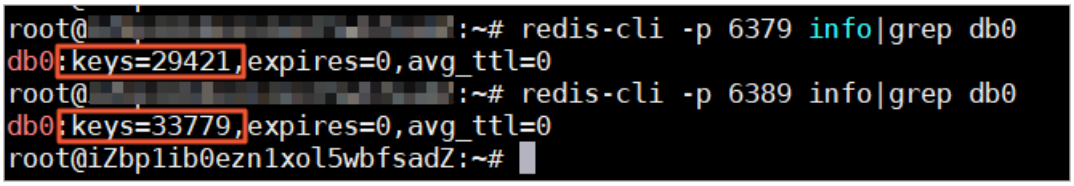


In this topic, the database DB0 is synchronized. You can use Data Management (DMS) to log on to the destination ApsaraDB for Redis instance and check the total number of keys in the ApsaraDB for Redis instance. The total number of keys is the same as that in the Twemproxy Redis cluster.

ApsaraDB for Redis instance



Source Twemproxy Redis cluster



1.2.7. Use an AOF file to migrate data

The redis-cli tool allows you to use an AOF file to migrate data from on-premises Redis to ApsaraDB for Redis.

The redis-cli tool is a native command-line tool of Redis. ApsaraDB for Redis allows you to use the redis-cli tool to seamlessly migrate existing Redis data to ApsaraDB for Redis. You can also use DTS to migrate data.

Precautions

- ApsaraDB for Redis can be accessed only from the Alibaba Cloud network. Therefore, you can use this migration solution only in Alibaba Cloud Elastic Compute Service (ECS) instances. If your on-premises Redis instance is not created in an Alibaba Cloud ECS instance, you must copy the existing AOF file to an ECS instance before importing data.
- The redis-cli tool is a native command-line tool of Redis. If you cannot use the redis-cli tool in an ECS instance, you can download and install Redis.

Procedure

For an on-premises Redis instance created in Alibaba Cloud ECS, migrate data as follows:

1. Enable the AOF feature of the existing on-premises Redis instance. You can skip this step if the AOF feature is enabled. Run the following command:

   ```
   # redis-cli -h old_instance_ip -p old_instance_port config set appendonly yes
   ```

2. Run the following command to import data from the AOF file, for example, the appendonly.aof file, into a new ApsaraDB for Redis instance:

   ```
   # redis-cli -h aliyun_redis_instance_ip -p 6379 -a password --pipe < appendonly.aof
   ```

   > **Notice**
   >
   > If your on-premises Redis instance does not need the AOF feature, you can run the following command to disable the feature after data is imported:

   ```
   # redis-cli -h old_instance_ip -p old_instance_port config set appendonly no
   ```

You can also watch the following video to learn how to migrate data from on-premises Redis deployed in ECS to ApsaraDB for Redis. The video lasts about 4 minutes.

1.3. Migrate between ApsaraDB Redis instances

1.3.1. Use redis-shake to migrate data

This topic describes how to use the rump mode of redis-shake to migrate data from an ApsaraDB for Redis instance to another ApsaraDB for Redis instance under the same Alibaba Cloud account.

Prerequisites

- An ApsaraDB for Redis instance is created as the destination of data migration.
- The destination instance supports the data structure or modules of the source instance.
- An Elastic Compute Service (ECS) instance is created for running the redis-shake tool.
- The IP address of the ECS instance is added to the whitelists of both the source and destination ApsaraDB for Redis instances.
- The ECS instance is running the Linux operating system.

Background

Redis-shake is an open-source tool developed by Alibaba Cloud. You can use it to parse (decode mode), recover (restore mode), back up (dump mode), and synchronize (sync/rump mode) Redis data. In rump mode, redis-shake can scan the source Redis to obtain full data and write the data to the destination Redis to migrate data. This migration solution does not use the SYNC or PSYNC command, and therefore has little impact on the service performance of Redis. It applies to Redis clusters and can be widely used to migrate data between on-premises Redis and ApsaraDB for Redis. This topic describes how to migrate data from an ApsaraDB for Redis instance to another ApsaraDB for Redis instance under the same Alibaba Cloud account.

For more information about the redis-shake tool, see redis-shake on GitHub or FAQ.

Warning The rump mode does not support incremental migration. We recommend that you stop writing data to the source Redis before migration to prevent data inconsistency.

Supported scenarios

- Supports data migration between different cloud products. In this case, either the source or destination must support Internet access.
- Supports data migration between versions. For example, you can migrate data from Redis 2.8 to Redis 4.0.
- Supports data migration from hybrid-storage instances to Community Edition instances or performance-enhanced instances.

Procedure

1. Log on to the ECS instance that can access both the source and destination ApsaraDB for Redis instances.
2. Download redis-shake on the ECS instance.

   Note We recommend that you download the latest version of redis-shake.

3. Run the following command to decompress the downloaded *redis-shake.tar.gz* package:

   ```
   # tar -xvf redis-shake.tar.gz
   ```

   Note In the decompressed folder, the *redis-shake* file is a binary file that can be run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of the redis-shake tool. You need to modify this configuration file in the next step.

4. Modify the redis-shake.conf file. The following table describes the parameters for the rump mode of the redis-shake tool.

Parameters for the rump mode of the redis-shake tool

Parameter	Description	Example
source.address	The connection address and service port of the source ApsaraDB for Redis instance.	r-bp1xxxxxxxxxxxx.redis.rds.aliyuncs.com

Parameter	Description	Example
source.password_raw	The password of the source ApsaraDB for Redis instance.	SourcePass233 Note If you use a database account other than the default database account to connect to the ApsaraDB for Redis instance, specify this parameter in the following format: account:password .
target.address	The connection address and service port of the destination ApsaraDB for Redis instance.	r-j6cxxxxxxxxxxxx.redis.rds.aliyuncs.com
target.password_raw	The password of the destination ApsaraDB for Redis instance.	TargetPass233
rewrite	Specifies whether to overwrite the data on the ApsaraDB for Redis instance with the data in the RDB file if the data has the same key. Valid values: ○ true: overwrites the data with the same key. ○ false: does not overwrite the data with the same key. Note Default value: true. We recommend that you back up the valid data of the destination ApsaraDB for Redis instance before you perform data migration. If you set this parameter to false and a key exists in both the source and destination databases, an error message is returned.	true
scan.key_number	The number of keys that redis-shake obtains each time it scans the source ApsaraDB for Redis instance. If you do not specify this parameter, the default value is set to 100.	100

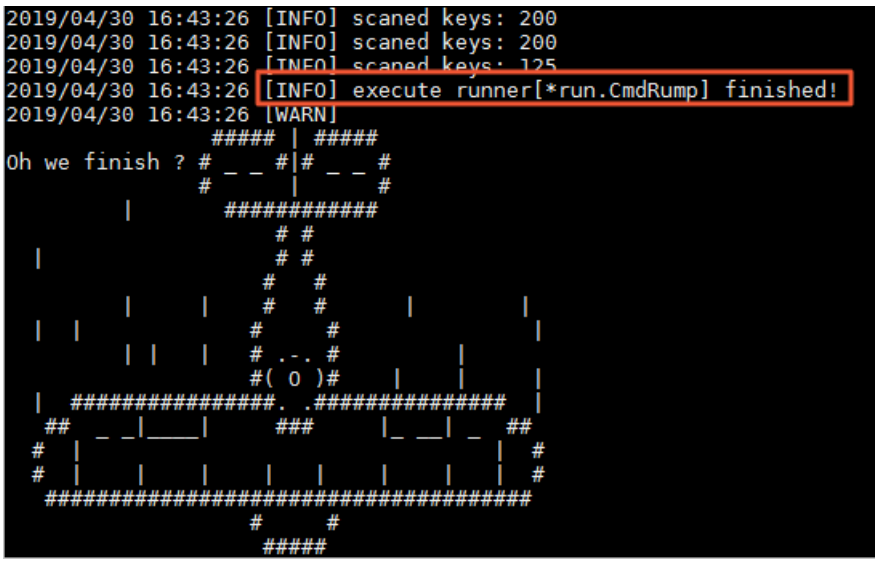
Parameter	Description	Example
scan.special_cloud	Specifies the specific edition of the source Redis instance.	aliyun_cluster Note This value specifies that the data is migrated from a source ApsaraDB for Redis cluster instance.

5. Run the following command to migrate data:

```
# ./redis-shake -type=rump -conf=redis-shake.conf
```

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you need to specify the correct file path in the command.

Migration example



Note When the message circled in red in the preceding figure appears, the data is migrated. Then, you can use the redis-full-check tool to check whether the data in the source and destination databases are consistent. For more information, see Verify data after migration.

1.4. Migrate data from a third-party database to ApsaraDB for Redis

1.4.1. Migrate data from Amazon ElastiCache for Redis to ApsaraDB for Redis

This topic describes how to migrate data from an AWS ElastiCache for Redis instance to an Alibaba Cloud ApsaraDB for Redis instance.

Prerequisites

- An ApsaraDB for Redis instance is created as the destination of data migration. For more information, see Create an ApsaraDB for Redis instance.
- An Elastic Compute Service (ECS) instance is created to run redis-shake. The 64-bit Linux operating system is running on the ECS instance. For more information, see Create an ECS instance.
- The ECS instance can access the destination ApsaraDB for Redis instance.

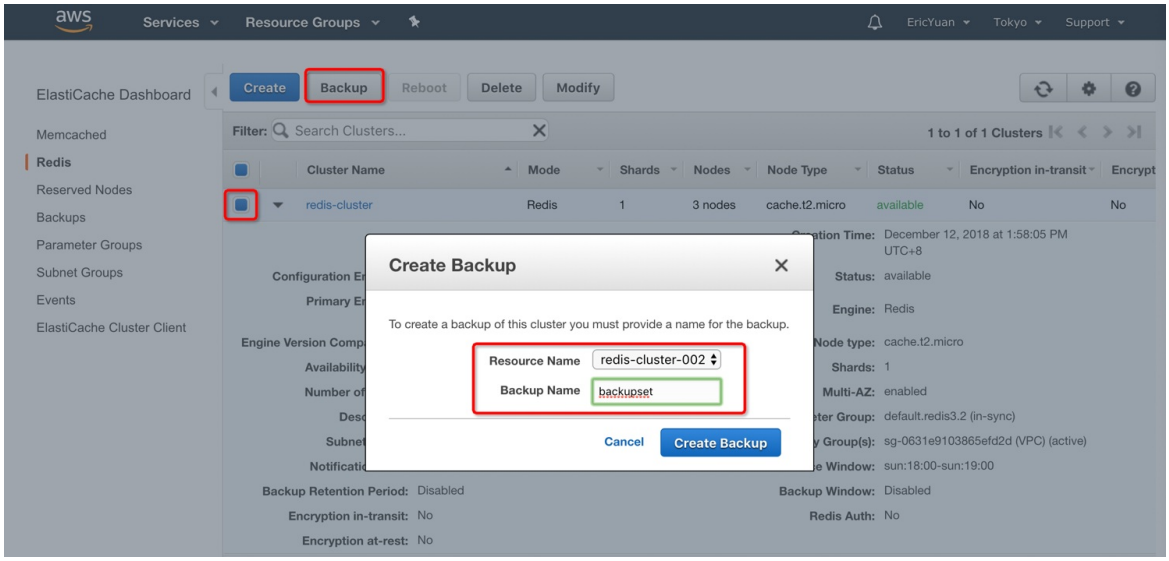
> Note
>
> - If the ECS instance and the destination ApsaraDB for Redis instance are deployed in the same Virtual Private Cloud (VPC), you must add the internal IP address of the ECS instance to the whitelist of the destination ApsaraDB for Redis instance. For more information, see Set IP address whitelists.
> - If the ECS instance and the destination ApsaraDB for Redis instance are not deployed in the same VPC network, the ECS instance can access the destination ApsaraDB for Redis instance by using the public endpoint. For more information, see Through the Internet.

Precautions

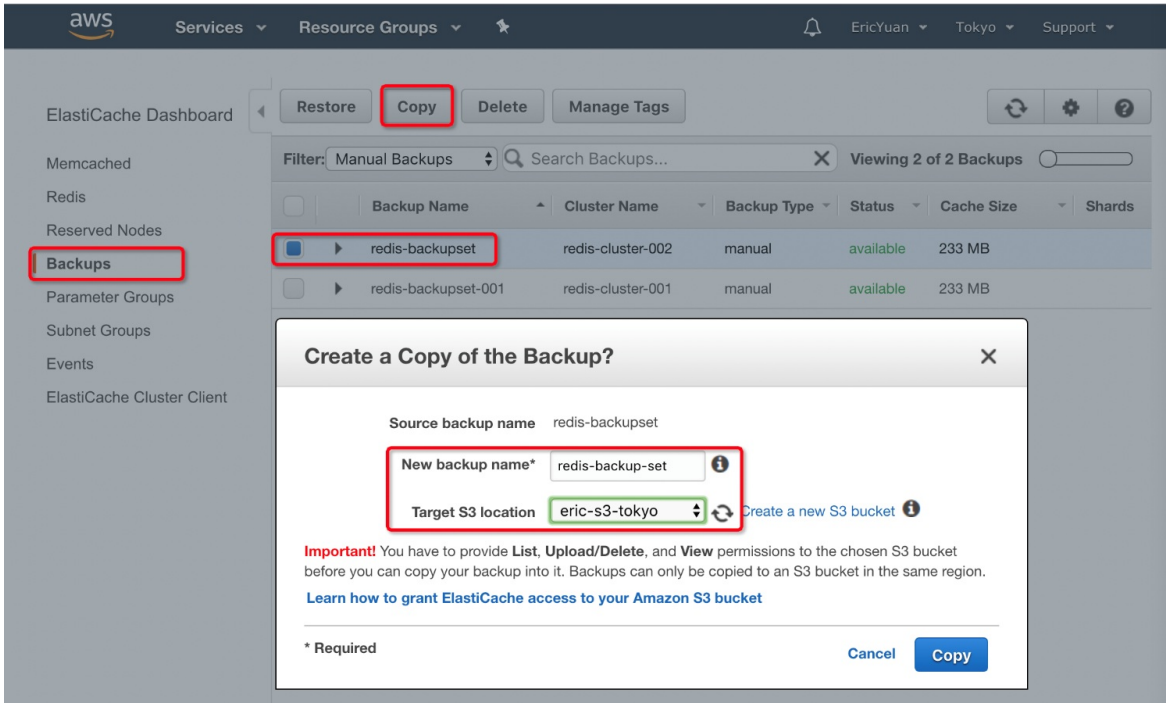
- We recommend that you stop writing data to the AWS ElastiCache for Redis instance before the migration.
- We recommend that you back up your data before the migration.
- We recommend that you schedule the service downtime before the migration.
- If you migrate data on an AWS ElastiCache for Redis instance that is created on an Amazon Elastic Compute Cloud (Amazon EC2) instance, we recommend that you use Alibaba Cloud Data Transmission Service (DTS) to migrate data.

Back up and export data from AWS ElastiCache for Redis to an S3 bucket

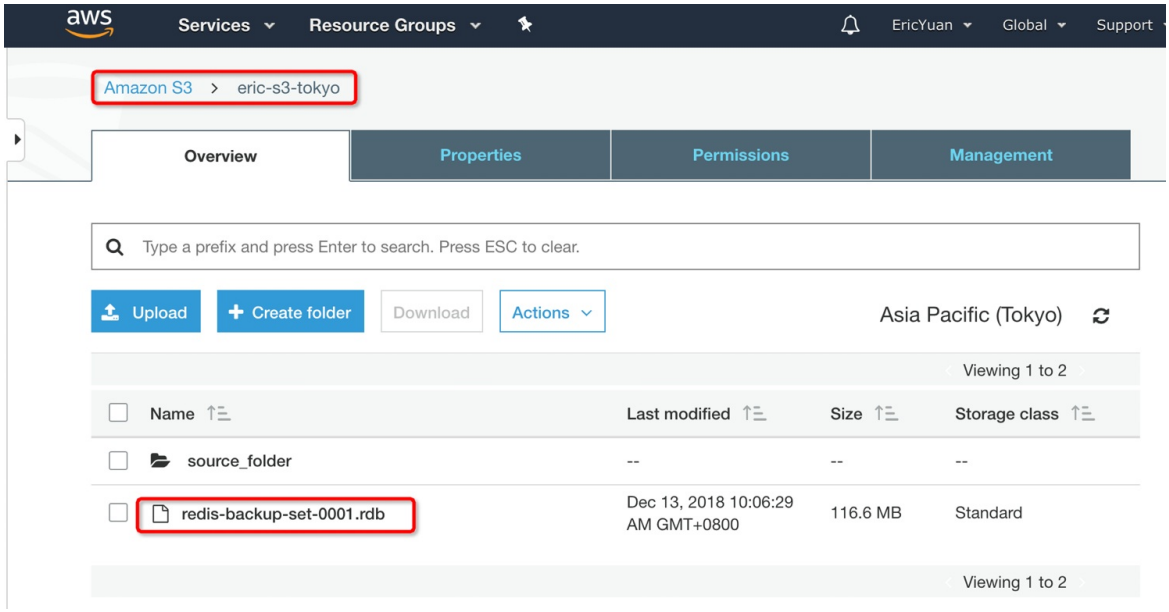
1. Create a backup for a specified cluster. Log on to the AWS ElastiCache console. In the left-side navigation pane, click **Redis**. Select the **cluster** that you want to back up, and click **Backup**. In the Create Backup dialog box that appears, select the **Resource Name** of the cluster, enter a **Backup Name**, and then click **Create Backup**.



2. Export backup files to an AWS S3 bucket. Select **Backups** on the left of the dashboard. Select the backup file created in the preceding step, click **Copy** above the backup file list. In the Create a Copy of the Backup dialog box that appears, enter a **New backup name**, select a **Target S3 location** and click **Copy** on the lower right.



3. You can find the exported RDB file in your S3 bucket.



4. Download the RDB file from the S3 bucket.

Procedure

1. Log on to the ECS instance that can access the destination ApsaraDB for Redis instance.
2. Download the RDB file of the source AWS ElastiCache for Redis instance to an ECS instance.

   > **Note**
   > - You can run the `wget <the download URL of the RDB file>` command to download the RDS file to an ECS instance.
   > - You can also use the MobaXterm Personal Edition client to upload the downloaded RDB file to an Alibaba Cloud ECS instance over SFTP.

3. Download redis-shake on the ECS instance.

   > **Note** We recommend that you download the latest version of redis-shake.

4. Run the following command to decompress the downloaded *redis-shake.tar.gz* package:

   ```
   # tar -xvf redis-shake.tar.gz
   ```

   > **Note** In the decompressed folder, the *redis-shake* file is a binary file that can run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of redis-shake. You must modify this configuration file.

5. Modify the *redis-shake.conf* file. The following table describes the parameters for the restore mode of redis-shake.

Parameters for the restore mode of redis-shake

Parameter	Description	Example
rdb.input	The path of the RDB file. You can specify either a relative path or an absolute path.	*/root/tools/RedisShake/demo.rdb*
target.address	The connection address and port of the destination ApsaraDB for Redis instance.	`r-bp1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379`
target.password_raw	The password of the destination ApsaraDB for Redis instance.	`TargetPass233` **Note** If you use a non-default database account to connect to the ApsaraDB for Redis instance, set this parameter in the format of `account:password`.
target.db	The database to which the data is recovered in the destination ApsaraDB for Redis instance. Default value: 0. For example, to recover the data of the source Amazon ElastiCache for Redis instance to db10 of the destination ApsaraDB for Redis instance, set this parameter to 10. If you set this parameter to a value less than 0, the data is recovered to db0.	`0`

Parameter	Description	Example
rewrite	Specifies whether to overwrite the data on the ApsaraDB for Redis instance with the data in the RDB file if the data has the same key. Valid values: ○ true: overwrites the data with the same key. ○ false: does not overwrite the data with the same key. Note Default value: true. We recommend that you back up the valid data of the destination ApsaraDB for Redis instance before you perform data migration. If you set this parameter to false and a key exists in both the source and destination databases, an error message is returned.	true
parallel	The number of concurrent threads used to synchronize the RDB file. You can increase the number of concurrent threads to improve the synchronization performance. Note ○ Minimum value: 1. ○ Maximum value: depends on the server performance. ○ Recommended value: 64.	64

Note You can use the default values for other parameters unless otherwise specified.

6. Run the following command to recover data:

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

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you must specify a valid file path in the command.

7. When `restore: rdb done` appears in logs, the data is recovered. You can press Ctrl+C to exit the tool.

Migration example

```
2019/04/26 17:56:37 [INFO] total = 11284825 -      2743474 [ 24%]  entry=11165
2019/04/26 17:56:38 [INFO] total = 11284825 -      5424236 [ 48%]  entry=23075
2019/04/26 17:56:39 [INFO] total = 11284825 -      8161432 [ 72%]  entry=35199
2019/04/26 17:56:40 [INFO] total = 11284825 -     10884277 [ 96%]  entry=47230
2019/04/26 17:56:40 [INFO] total = 11284825 -     11284825 [100%]  entry=50002
2019/04/26 17:56:40 [INFO] restore: rdb done
2019/04/26 17:56:40 [INFO] Enabled http stats, set status (incr), and wait forever.
```

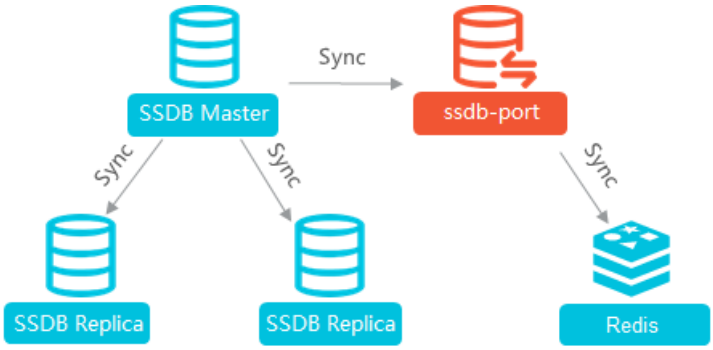
1.4.2. Migrate data from SSDB to ApsaraDB for Redis

You can use the ssdb-port tool to migrate data from SSDB to ApsaraDB for Redis.

Background

How it works

As an SSDB replica node, the ssdb-port tool synchronizes data from the SSDB master node that serves as the source database. Then, the ssdb-port tool parses and converts the data to the format supported by ApsaraDB for Redis, and sends the data to the ApsaraDB for Redis instance that is specified in the configuration file. The following figure shows the migration process.



After full synchronization is completed, incremental data in the SSDB database is synchronized to the ApsaraDB for Redis instance before ssdb-port is disconnected from the instance.

> **Note**
>
> - If you run a command that is not supported by the ssdb-port tool to modify data in the source SSDB database, the modified data cannot be synchronized to the ApsaraDB for Redis instance. For more information about the SSDB commands that are supported by the ssdb-port tool, see the following list.
> - If you need to synchronize more commands, log on to the Connect Platform to propose a suggestion.

Commands supported by the ssdb-port tool

- set
- setx
- setnx
- expire
- del
- get
- incr

- qpop_front
- qpush_front
- qclear
- qtrim_front
- qtrim_back
- zset
- zdel
- zincr
- multi_zdel
- multi_zset
- hset
- hdel
- hclear
- multi_hset
- multi_hdel
- hincr

Prerequisites

- A Linux-based Elastic Compute Service (ECS) instance is created in the same Virtual Private Cloud (VPC) as the destination ApsaraDB for Redis instance, and can connect to the destination ApsaraDB for Redis instance.
- The version of the source SSDB database is 1.9.2 or later.

Procedure

1. Run the following commands in the ECS instance to download and decompress the ssdb-port.tar.gz package:

```
# wget http://docs-aliyun.cn-hangzhou.oss.aliyun-inc.com/assets/attach/94155/cn_zh/1547627852086/ssdb-port.tar.gz
# tar -xvf ssdb-port.tar.gz
# cd ssdb-port
```

2. Run the following command to modify the configuration file of the ssdb-port tool based on the example:

```
vi ssdb_port.conf
```

The following code shows an example of the *ssdb_port.conf* file. Modify the connection information of the source SSDB database and the destination ApsaraDB for Redis instance based on the comments.

```
# SSDB server configuration for replication
# You MUST indent the code with the Tab key.


# The relative path of this file. The directory must exist.
```

```
work_dir = ./var_ssdb_port
pidfile = ./var_ssdb_port/ssdb.pid

# The connection information of the ssdb-port tool, which does not need to be modified.
server:
  ip: 127.0.0.1
  port: 8890
  #readonly: yes

replication:
  binlog: yes
    capacity: 100000000
  # The maximum synchronization speed. Unit: MB/s. A value of -1 indicates that the speed is not limited.
  sync_speed: -1
  slaveof:
    # The ID of the master node. This ID takes effect even when the IP address or port number of the master node is changed.
    # If this parameter is left empty or not specified, the IP address and port number (in the form at of IP address:port) will be used.
    id: svc_1
    # The replication type. Valid values: sync and mirror. Default value: sync.
    type: sync
    host: localhost # The connection address of the SSDB master node (the source SSDB database).
    port: 8888 # The port of the SSDB master node (the source SSDB database).
    #auth: password
    redis_host: localhost # The connection address of the destination ApsaraDB for Redis instance.
    redis_port: 6379 # The port number of the destination ApsaraDB for Redis instance.
    redis_auth: password # The password of the destination ApsaraDB for Redis instance.

logger:
  level: debug
  output: log_ssdb_port.txt
  rotate:
    size: 1000000000

leveldb:
  # Unit: MB.
  cache_size: 500
```

```
# Unit: MB.
write_buffer_size: 64
# Unit: MB/s.
compaction_speed: 1000
# Compression flag. Value values: yes and no.
compression: yes
```

3. Run the `./ssdb-port-2.17 ssdb_port.conf` command to start synchronization.
4. Connect to the ApsaraDB for Redis instance to check whether data synchronization is completed.

> **Note** You can use the redis-cli tool or Data Management Service (DMS) to connect to the ApsaraDB for Redis instance. For more information about how to connect to the instance, see Quick Start.

> **Notice** If you run the hset or hget command to modify data, the modified data cannot be synchronized if the target keys are in Chinese. This restriction does not apply to other supported commands.

1.4.3. Migrate data from Google Cloud Memorystore for Redis to ApsaraDB for Redis

You can use the rump tool to migrate data from Google Cloud Memorystore for Redis to ApsaraDB for Redis.

Prerequisites

- An Elastic Compute Service (ECS) instance and an ApsaraDB for Redis instance are created in Alibaba Cloud, and they can communicate with each other through the internal network.
- A public endpoint is configured so that you can access the ApsaraDB for Redis instance from the ECS instance over the Internet.

  > **Notice** The public connection address is used only for data migration through the Internet. After data is migrated, clear the related public connection address and only access the ApsaraDB for Redis instance through the internal network to ensure data security.

- The rump tool is downloaded in Google Compute Engine, and the owner of this file is granted the execute permission.

How it works

The rump tool uses the **SCAN** command to obtain the key list from the source Cloud Memorystore for Redis instance, and uses the **DUMP** command to retrieve the key content. Then, it uses the **PTTL** command to obtain the expiration time, and uses the **RESTORE** command to synchronize the keys to the destination instance through pipelines.

> **Note** The rump tool does not support incremental migration.

Procedure

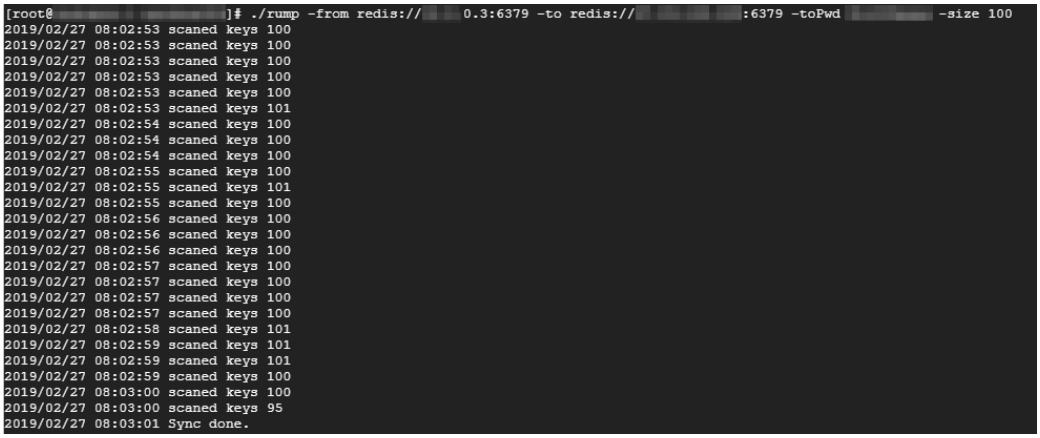
1. Run the following command in Google Compute Engine to migrate data:

```
./rump -from source_addr -fromPwd source_pwd -to dest_addr -toPwd dest_pwd [-size size] [-replace]
```

Parameters for the rump tool

Parameter	Description
source_addr	The address of the source Cloud Memorystore for Redis instance, in the format of `redis://host:port/db`. Set this parameter to the private IP address of the Cloud Memorystore for Redis instance. The host and port fields are required. If the db field is not set, the default value 0 is used.
source_pwd	The password of the source Cloud Memorystore for Redis instance. You do not need to set this parameter if the Cloud Memorystore for Redis instance does not have a password.
dest_addr	The address of the destination ApsaraDB for Redis instance, in the format of `redis://host:port/db`. Set this parameter to the public connection address of the ApsaraDB for Redis instance. The host and port fields are required. If the db field is not set, the default value 0 is used.
dest_pwd	The password of the destination ApsaraDB for Redis instance.
size	The number of keys scanned and synchronized at a time. Default value: 10.
replace	Specifies whether to overwrite the existing keys in the destination ApsaraDB for Redis instance that are identical to those synchronized from the source Cloud Memorystore for Redis instance. If you do not set this parameter and any keys are duplicate in the source and destination databases, an error message is returned. **Note** If you set this parameter, make sure that you have backed up important data in the destination ApsaraDB for Redis instance to avoid data loss after overwriting.

Migration example



2. Check whether all data is migrated to the ApsaraDB for Redis instance.

1.5. Migrate from ApsaraDB Redis to on-premises Redis

1.5.1. Use a backup file to migrate data

You can use the restore mode of the redis-shake tool to migrate data from ApsaraDB for Redis to on-premises Redis through a backup file.

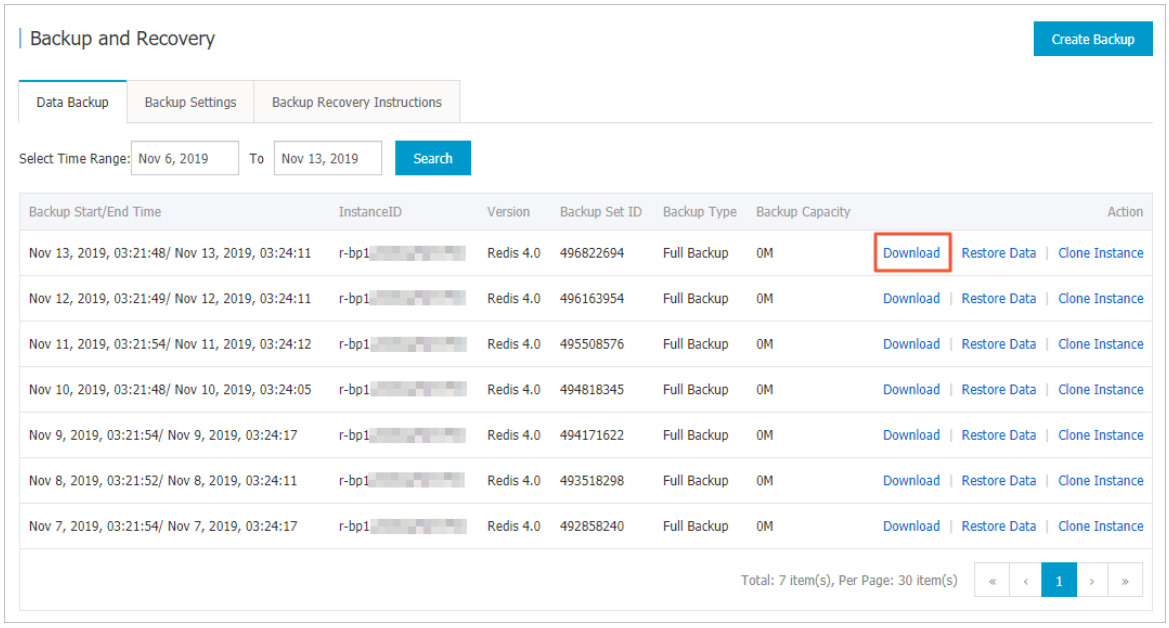
Prerequisites

- An on-premises Redis instance is created as the destination of data migration.
- The 64-bit Linux operating system is running on the server for running the redis-shake tool.
- The Linux server can access the on-premises Redis instance.
- The data of the source ApsaraDB for Redis instance is backed up in the console. For more information, see Back up and restore data in the console.

Download a backup file in the ApsaraDB for Redis console

1. Log on to the ApsaraDB for Redis console.
2. In the top navigation bar, select the region where the instance resides.
3. On the Instance List page, find the target instance. Then, click the instance ID or click **Manage** in the **Actions** column.
4. In the left-side navigation pane, click **Backup and Recovery**.
5. On the **Backup and Recovery** page, find the backup file to migrate and click **Download** in the Actions column.

> **Note** A backup file is generated for each node of an ApsaraDB for Redis cluster.



Download the redis-shake tool

Download the redis-shake tool.

> **Note** We recommend that you download the latest version.

Run the following command to decompress the *redis-shake.tar.gz* package:

```
# tar -xvf redis-shake.tar.gz
```

> **Note** In the decompressed folder, the *redis-shake* file is a binary file that can run in the 64-bit Linux operating system. The *redis-shake.conf* file is the configuration file of the redis-shake tool. You must modify this configuration file in the next step.

Use a backup file to migrate data

1. Modify the *redis-shake.conf* file. The following table describes the parameters for the restore mode of the redis-shake tool.

Parameters for the restore mode of the redis-shake tool

Parameter	Description	Example
rdb.input	The path of the RDB file. You can specify either a relative path or an absolute path.	*/root/tools/RedisShake/demo.rdb*
target.address	The endpoint and port of the destination on-premises Redis instance.	r-bp1xxxxxxxxxxxxx.redis.rds.aliyuncs.com:6379

Parameter	Description	Example
target.password_raw	The password of the destination on-premises Redis instance.	`TargetPass233` Note If you use a database account other than the default database account to connect to the on-premises Redis instance, set this parameter in the following format: `account:password`.
target.db	The database to which the data is recovered in the destination on-premises Redis instance. Default value: 0. For example, to recover the data of the source ApsaraDB for Redis instance to DB10 of the destination on-premises Redis instance, set this parameter to 10. If you set this parameter to a value less than 0, data is recovered to DB0.	`0`
rewrite	Specifies whether to overwrite the existing keys that are identical to those in the RDB file. Valid values: ○ true ○ false Note Default value: true. We recommend that you back up the valid data of the destination on-premises Redis instance before migration. If you set this parameter to false and any keys are duplicate in the source and destination databases, an error message is returned.	`true`

Parameter	Description	Example
parallel	The number of concurrent threads used to synchronize the RDB file. More concurrent threads improve synchronization performance. **Note** ○ Minimum value: 1. ○ Maximum value: depends on the server performance. ○ Recommended value: 64.	64

Note You can use the default values for other parameters unless otherwise specified.

2. Run the following command to recover data:

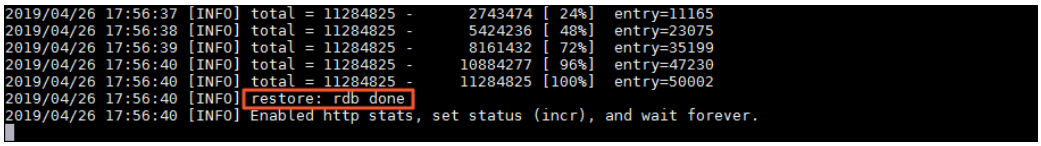
```
# ./redis-shake -type=restore -conf=redis-shake.conf
```

Note You must run this command in the same directory as the *redis-shake* and *redis-shake.conf* files. Otherwise, you must specify the correct file path in the command.

3. When `restore: rdb done` appears in logs, the data is recovered. You can press Ctrl+C to exit the tool.

Note You must repeat the preceding procedure to migrate the backup data of all nodes in an ApsaraDB for Redis cluster.

Example



1.6. Verify data after migration

After Redis data is migrated, you must check whether the data is consistent between the source and destination instances.

Prerequisites

- Redis data is migrated.

Note For more information about how to migrate data in ApsaraDB for Redis, see Overview.

- The source and destination Redis instances are in the following architectures: master-replica, standalone, open-source cluster, ApsaraDB for Redis cluster with proxy nodes, and TencentDB for Redis cluster with proxy nodes.
- A Linux-based Elastic Compute Service (ECS) instance is created for running the redis-full-check tool. For more information, see Create an ECS instance.
- The ECS instance can access the source and destination Redis instances.

> **Note**
> - If the ECS instance and the source or destination Redis instance are in the same Virtual Private Cloud (VPC), you must add the internal IP address of the ECS instance to the whitelist of the Redis instance. For more information, see Set IP address whitelists.
> - If the ECS instance and the source or destination Redis instance are not in the same VPC, the ECS instance can access the Redis instance through the public endpoint. For more information, see Through the Internet.

Introduction to the redis-full-check tool

If an exception occurs during Redis data migration, the data is inconsistent between the source and destination Redis instances. You can use the redis-full-check tool to find the inconsistent data.

The redis-full-check tool is a Redis data verification tool developed by Alibaba Cloud. It can extract data from the source and destination instances, compare them for multiple times, and then record the comparison results in an SQLite3 database. This tool can be used to verify full data.

> **Note** For more information about the redis-full-check tool, see redis-full-check on GitHub.

Procedure

1. Log on to the ECS instance. For more information, see Connect to an ECS instance.
2. Download the redis-full-check tool on the ECS instance.

   > **Note** We recommend that you download the latest version.

3. Run the following command to decompress the *redis-full-check.tar.gz* package:

   ```
   tar -xvf redis-full-check.tar.gz
   ```

4. Run the following command to verify data:

   ```
   ./redis-full-check -s "<Endpoint 1 of the source Redis cluster:Port of endpoint 1;Endpoint 2 of the source Redis cluster:Port of endpoint 2;Endpoint 3 of the source Redis cluster:Port of endpoint 3;>" -p <Password of the source Redis cluster> -t <Endpoint of the destination Redis instance:Port> -a <Password of the destination Redis instance> --comparemode=1 --comparetimes=1 --qps=10 --batchcount=100 --sourcedbtype=1 --targetdbfilterlist=0
   ```

The following table describes common parameters of the redis-full-check tool. For more information, see Configure redis-full-check.

Common parameters of redis-full-check

Parameter	Description	Example
-s	The endpoint and port of the source Redis instance. Note ◦ If the source Redis instance is a cluster, separate cluster endpoints with semicolons (;). ◦ Enclose the cluster endpoints in a pair of double quotation marks ("). ◦ This parameter is required.	r-bp1xxxxxxxxxxxx.redis.rds.aliyuncs.com:6379 "10.xx.xx.1:7000;10.xx.xx.1:7001;10.xx.xx.2:7002;10.xx.xx.2:7003"
-p	The password of the source Redis instance.	SourcePwd233

Parameter	Description	Example
-t	The endpoint and port of the destination Redis instance. **Note** ◦ If the destination Redis instance is a cluster, separate cluster endpoints with semicolons (;). ◦ Enclose the cluster endpoints in a pair of double quotation marks ("). ◦ This parameter is required.	r-bp1xxxxxxxxxxxxx.redis.rds.aliyuncs.com:6379 "10.xx.xx.1:7000;10.xx.xx.1:7001;10.xx.xx.2:7002;10.xx.xx.2:7003"
-a	The password of the destination Redis instance.	TargetPwd233
--sourcedbtype	The type of the source Redis instance. Valid values: ◦ 0: standalone or master-replica. ◦ 1: cluster. ◦ 2: ApsaraDB for Redis or TencentDB for Redis.	--sourcedbtype=1

Parameter	Description	Example
--sourcedbfilterlist	The databases for which you want to verify data in the source Redis instance. Note ◦ This parameter is not required for open-source Redis clusters. ◦ If you do not set this parameter, the data in all databases is verified. ◦ Separate multiple databases with semicolons (;).	--sourcedbfilterlist=0;1;2
--targetdbtype	The type of the destination Redis instance. Valid values: ◦ 0: standalone or master-replica. ◦ 1: cluster. ◦ 2: ApsaraDB for Redis or TencentDB for Redis.	--targetdbtype=0

Parameter	Description	Example
--targetdbfilterlist	The databases for which you want to verify data in the destination Redis instance. **Note** ◦ This parameter is not required for open-source Redis clusters. ◦ If you do not set this parameter, the data in all databases is verified. ◦ Separate multiple databases with semicolons (;).	`--targetdbfilterlist=0;1;2`
-d	The name of the file for storing inconsistent keys. Default value: result.db.	`xxx.db`
--comparetimes	The number of times that the data is verified. ◦ Default value: 3. ◦ Minimum value: 1. ◦ We recommend that you set this parameter to a value that does not exceed 5.	--comparetimes=1
-m	The verification mode. Valid values: ◦ 1: verifies full data. ◦ 2: only verifies the length of the value. ◦ 3: only checks whether the keys exist. ◦ 4: verifies full data except for big keys.	1

Parameter	Description	Example
--qps	The queries per second (QPS). Note ○ Minimum value: 1. ○ Maximum value: depends on the server performance.	--qps=10
--filterlist	The keys to verify. Separate multiple keys with vertical bars (|). Note ○ *abc**: matches all keys that start with abc. ○ *abc*: matches the abc key.	--filterlist=abc*|efg|m*

Note After the verification is completed, the comparison result appears on the CLI. For example, the following result indicates that two keys are inconsistent between the source and destination Redis instances. If the number of inconsistent keys is 0, the data is consistent between the source and destination Redis instances.

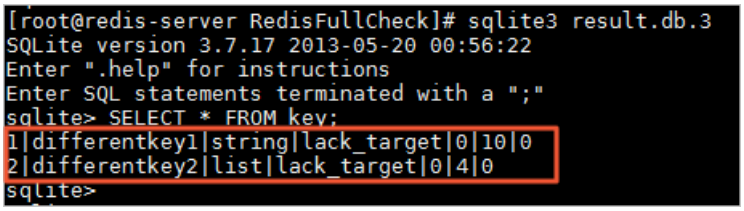
```
all finish successfully, totally 2 keys or fields conflict
```

5. Query the inconsistent keys in the SQLite3 database.
   i. Run the `sqlite3 result.db.3` command.

Note Inconsistent keys are stored in result.db.3 by default.

ii. Run the `SELECT * FROM key;` statement.

Query inconsistent keys



Note The SQLite3 database provides the key and field tables.

- The key table stores the inconsistent keys.
- The field table stores details about inconsistent data of the hash, set, zset, and list types.

2.Data Synchronization

2.1. Overview

ApsaraDB for Redis provides multiple data synchronization solutions for you to synchronize data in different scenarios.

Synchronization tools

ApsaraDB for Redis provides two tools for data synchronization: Data Transmission Service (DTS) and redis-shake.

Tool	Description
DTS	Alibaba Cloud DTS is a real-time data streaming service. The service allows you to migrate, subscribe to, and synchronize data by using stable and secure transmission links. DTS provides an API and a visualized console to enable efficient operations. We recommend that you use DTS for data synchronization. For more information, see What is DTS?.
redis-shake	Redis-shake is an open source Linux-based tool developed by Alibaba Cloud. You can use this flexible and efficient tool to parse (decode mode), recover (restore mode), back up (dump mode), and synchronize (sync or rump mode) data on ApsaraDB for Redis instances. If DTS is unavailable, you can use redis-shake to synchronize data.

Scenarios

Scenario	Synchronization tool	Solution
Synchronize data from an on-premises database to an ApsaraDB for Redis instance	DTS	• Configure one-way data synchronization between ApsaraDB for Redis instances Note The on-premises Redis database must be connected to Alibaba Cloud over leased lines, VPN gateways, or Smart Access Gateway (SAG) services. • Synchronize data from a user-created Redis cluster to an ApsaraDB for Redis cluster instance
	redis-shake	• Use redis-shake to synchronize data from one ApsaraDB for Redis instance to another

Scenario	Synchronization tool	Solution
	ssdb-port	• Use ssdb-port to synchronize data from SSDB to ApsaraDB for Redis Note You can use ssdb-port to synchronize data from Sequence Similarity DataBase (SSDB) to ApsaraDB for Redis in real time.
Synchronize data from a user-defined database on an Elastic Compute Service (ECS) instance to an ApsaraDB for Redis instance	DTS	• Synchronize data from a user-created Redis database hosted on ECS to an ApsaraDB for Redis instance • Synchronize data from a Codis cluster hosted on ECS to an ApsaraDB for Redis instance • Synchronize data from a Twemproxy Redis cluster hosted on ECS to an ApsaraDB for Redis instance
	redis-shake	• Use redis-shake to synchronize data from one ApsaraDB for Redis instance to another
Synchronize data among ApsaraDB for Redis instances	DTS	• Configure two-way data synchronization between ApsaraDB for Redis Enhanced Edition instances • Configure one-way data synchronization between ApsaraDB for Redis instances
	redis-shake	• Use redis-shake to synchronize data from one ApsaraDB for Redis instance to another
Synchronize data from an on-premises Redis database to a user-defined Redis database that is hosted on an ECS instance	DTS	• Synchronize data from a user-created Redis database connected over Express Connect, VPN Gateway, or Smart Access Gateway to a user-created Redis database hosted on ECS
	redis-shake	• Use redis-shake to synchronize data from one ApsaraDB for Redis instance to another