

TencentDB for Redis®

Command Reference

Product Documentation



Copyright Notice

©2013–2026 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Command Reference

- Command Reference Overview

- Redis Edition Compatible Command List

 - Overview of Commands Supported in the Community Edition

 - Connection Group

 - Hash Group Keys

 - Keys

 - List Group

 - Pub Group and Sub Group

 - Sets Group

 - Sorted Sets Group

 - Strings Group

 - Transactions

 - Hyperloglog Group

 - scripting and functions Group

 - Geo Group

 - Server Group

 - Stream Group

- Differences in Redis Major Version Command Usage

- Differences Between the Proxy Architecture and Direct Connection Mode

- More Command Operations (Redis/Valkey Edition)

 - List of Supported DMC Commands

 - Redis/Valkey Edition Partially Supported Command Usage Examples

 - Use Cases of Custom Commands

- Memcached Edition Command Compatibility

Command Reference

Command Reference Overview

Last updated: 2026-03-17 20:46:27

Cluster Architecture stores data in a distributed manner, and its biggest difference from standard architecture lies in whether a single command supports multikey access. For the cluster architecture, commands can be categorized into partially supported, custom, and unsupported as detailed below:

Note:

- The current Redis 7.0 and 6.2 versions do not support the RESP3 protocol.
- The Valkey edition is fully compatible with all Redis 7.0 commands, ensuring seamless migration without requiring any modifications to existing application code.

Command Type	Description
Unsupported commands	This refers to commands from community editions that are not supported by Tencent Cloud Distributed Cache, which returns <code>(error) ERR unknown command 'keys'</code> message. The supported commands vary by version. For more information on the commands supported by different versions and architectures, see Overview .
Partially supported commands	Cluster Architecture is compatible with smart clients such as JedisCluster. For compatibility with JedisCluster, Tencent Cloud Distributed Cache modifies the IP list returned by the supported commands, and the IP address of each node in the returned information is the instance's VIP. For more information, see Use Case of Partially Supported Commands .
Supported cross-slot commands	The in-memory version (cluster architecture) currently supports cross-slot access commands, including MGET, MSET, DEL, Hset, Scan, Keys. However, cross-slot access for other multi-KEY commands is not yet supported.
Custom commands	Custom commands support the access of each node in a cluster. A new parameter Node ID is added to the right of the parameter list of the original command, including <code>INFO</code> , <code>MEMORY</code> , <code>SLOWLOG</code> , <code>FLUSHDB</code> , <code>PING</code> , and <code>KEYS</code> (with hashtag supported for preferred match). For more information, see Use Cases of Custom Commands .
Supported DMC commands	Database Management Center (DMC) allows you to log in to your TencentDB instances to access them, view their key metric information, and run Redis commands. For more information, see List of Supported DMC Commands .

Transactional commands	Cluster Architecture supports transactional commands provided that the transactions are started by the <code>WATCH</code> command. The keys of a transaction should be stored in the same slot, and the keys of <code>WATCH</code> and transaction-related keys should also be stored in the same slot. Hashtag is recommended for multikey transactions in cluster mode.
Multi-database commands	Cluster Architecture supports multiple databases (256 by default); therefore, it can support all commands related to database operations.

Redis Edition Compatible Command List

Overview of Commands Supported in the Community Edition

Last updated: 2025-12-23 17:43:29

For supported commands by versions and architecture, see the following command groups. In the command group tables, ✓ indicates "supported", x indicates "unsupported", and – indicates that cross-slot access is not applicable to the command.

- [Connection Group](#)
- [Hash Group](#)
- [Keys](#)
- [list group](#)
- [pub/sub group](#)
- [sets group](#)
- [sorted sets group](#)
- [strings group](#)
- [transactions group](#)
- [hyperloglog group](#)
- [scripting group](#)
- [geo group](#)
- [server group](#)
- [stream group](#)

Note:

For detailed descriptions of command parameters and usage examples, see [Redis Command Reference](#).

Connection Group

Last updated: 2026-03-17 18:10:58

- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the auth, echo, ping, quit, and select commands.
- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) do not support the client caching, client getredir, client info, client tracking, client trackinginfo, client unpause, reset, and client list commands.
- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 Cluster Architecture do not support the swapdb command, whereas Redis versions 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 Standard Architecture support swapdb.
- The hello command was added for compatibility with Redis version 6.2 and currently only supports the RESP2 protocol.
- The auth, echo, ping, quit, select, swapdb, and hello commands do not have cross-slot scenarios (Cluster Architecture).

For detailed information on the versions supporting the auth, echo, ping, quit, select, swapdb, and hello commands in the connection family, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve scenarios of cross-slot access.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Support for Cross-Slot
auth	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
auth name pwd	✓	✓	✓	✓	x	x	x	x	x	-
echo	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ping	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
quit	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
select	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
swapdb	✓	x	✓	x	✓	x	✓	x	x	-
hello	✓	✓	✓	✓	x	x	x	x	x	-
client caching	x	x	x	x	x	x	x	x	x	-
client getredir	x	x	x	x	x	x	x	x	x	-
client info	x	x	x	x	x	x	x	x	x	-
client tracking	x	x	x	x	x	x	x	x	x	-
client trackinginfo	x	x	x	x	x	x	x	x	x	-
client unpause	x	x	x	x	x	x	x	x	x	-
reset	x	x	x	x	x	x	x	x	x	-
client list	x	x	x	x	x	x	x	x	x	-
client kill	✓	✓	✓	✓	x	x	x	x	x	-

Hash Group Keys

Last updated: 2026-03-17 18:10:58

Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the hdel, hexists, hget, hgetall, hincrby, hincrbyfloat, hkeys, hlen, hmget, hmset, hset, hsetnx, hstrlen, hvals, and hscan commands. Only versions 6.2, 7.0, and ValKey 8.0 support the hrandfield command. Hash commands are not supported in cross-slot scenarios for cluster architectures. For details, see the table below, where ✓ indicates support, x indicates no support, and – indicates that the command has no cross-slot access scenario.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support for Cluster Architecture
hdel	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hexists	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hget	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hgetall	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hincrby	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hincrbyfloat	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hkeys	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hlen	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hmget	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hmset	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hset	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hsetnx	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hstrlen	✓	✓	✓	✓	✓	✓	✓	✓	x	–
hvals	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hscan	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
hrandfield	✓	✓	✓	✓	x	x	x	x	x	–

Keys

Last updated: 2026-03-17 18:10:58

- Redis 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 Standard Architecture and Cluster Architecture all support the touch, restore, object, unlink, wait, dump, del, scan, exists, expire, expireat, keys, type, move, ttl, persist, pexpire, pexpireat, pttl, randomkey, rename, renamenx, and sort commands.
- Only Redis 6.2, 7.0, and its ValKey 8.0 support the copy command and object help command.
- Cluster Architecture supports the del and exists commands in cross-slot scenarios, but does not support unlink, rename, or renamenx.
- Redis 7.0 version and its ValKey 8.0 support: sort_ro, expiretime, pexpiretime.

Command Family	Command	New Description
Keys Family	EXPIRETIME key	<ul style="list-style-type: none"> • New commands added. • Returns the second-precision expiration timestamp of the key.
	PEXPIRETIME key	<ul style="list-style-type: none"> • New commands added. • Returns the millisecond-precision expiration timestamp of the key.
	EXPIRE key seconds [NX XX GT LT]	<p>New parameters added: NX XX GT LT.</p> <ul style="list-style-type: none"> • NX sets an expiration time only when the key does not have an expiration time. • XX sets an expiration time only when the key has an expiration time. • GT sets a new expiration time only when the new expiration time is greater than the current expiration time. • LT sets a new expiration time only when the new expiration time is less than the current expiration time.
	SORT_RO key [BY pattern] [LIMIT offset count] [GET pattern [GET pattern ...]] [ASC DESC] [ALPHA]	<ul style="list-style-type: none"> • New commands added. • Sort the sorted set (sorted set) in Redis and obtain the sorted results.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve scenarios of cross-slot access.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Cross-Slot Support
touch	✓	✓	✓	✓	✓	✓	✓	✓	x	-
restore	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
object	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
object help	✓	✓	✓	✓	x	x	x	x	x	-
unlink	✓	✓	✓	✓	✓	✓	✓	✓	x	x
wait	✓	✓	✓	✓	✓	✓	✓	✓	x	-
migrate	x	x	x	x	x	x	x	x	x	-
dump	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
del	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
scan	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
exists	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
expire	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
expireat	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
keys	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
type	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
move	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ttl	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
persist	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
pexpire	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
pexpireat	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
pttl	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
randomkey	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
rename	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
renamenx	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sort	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
sort_ro	✓	✓	x	x	x	x	x	x	x	-
copy	✓	✓	✓	✓	x	x	x	x	x	-
expiretime	✓	✓	x	x	x	x	x	x	x	-
pexpiretime	✓	✓	x	x	x	x	x	x	x	-

List Group

Last updated: 2026-03-17 18:10:58

- Redis version 2.8 Standard Architecture, Redis 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 Standard Architecture and Cluster Architecture all support lindex, linsert, llen, lpop, lpush, lpushx, lrange, lrem, lset, ltrim, rpop, rpoplpush, rpush, rpushx, bpop, brpop, brpoplpush commands.
- The Standard Architecture of Redis version 2.8, Redis 4.0, and Redis 5.0 do not support the blmove, lmove, and lpos commands, whereas Redis 6.2 and 7.0 do.
- Cluster Architecture does not support the rpoplpush, bpop, brpop, or brpoplpush commands in cross-slot scenarios.
- Redis 7.0 version and ValKey 8.0 support: lmpop, blmpop.

Command Family	Command	New Description
list Family	LMPOP numkeys key [key ...] <LEFT RIGHT> [COUNT count]	<ul style="list-style-type: none"> • New commands added. • Pop COUNT elements from the list, with the popping direction controlled by LEFT RIGHT. • Multi-key commands.
	BLMPOP timeout numkeys key [key ...] <LEFT RIGHT> [COUNT count]	<ul style="list-style-type: none"> • New commands added. • The blocking version of lmpop, where timeout is used to set the blocking time. • If blmpop is used in multi exec, its behavior is equivalent to lmpop. This specification is defined by Redis, and proxy does not perform special handling.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Cross-Slot Support
lindex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
linsert	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
llen	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lpop	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lpush	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lpushx	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lrange	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lrem	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
lset	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ltrim	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
rpop	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
rpoplpush	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
rpush	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
rpushx	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
blpop	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
brpop	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
brpoplpush	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
blmove	✓	✓	✓	✓	x	x	x	x	x	-
lpos	✓	✓	✓	✓	x	x	x	x	x	-
lmove	✓	✓	✓	✓	x	x	x	x	x	-
lmpop	✓	✓	x	x	x	x	x	x	x	-
blmpop	✓	✓	x	x	x	x	x	x	x	-

Pub Group and Sub Group

Last updated: 2026-03-17 18:10:58

Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the psubscribe, pubsub, publish, punsubscribe, subscribe, and unsubscribe commands.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Support for Cross-Slot
psubscribe	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
pubsub	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
publish	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
punsubscribe	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
subscribe	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
unsubscribe	✓	✓	✓	✓	✓	✓	✓	✓	✓	-

Sets Group

Last updated: 2026-03-17 18:10:58

- Redis 2.8 Standard Architecture, Redis 4.0, and 5.0 (both Standard Architecture and Cluster Architecture) support the sadd, scard, sdiff, sdiffstore, sinter, sinterstore, sismember, smembers, smove, spop, srandmember, srem, sscan, sunion, and sunionstore commands, but do not support the smismember command.
- Redis 6.2, 7.0, and ValKey 8.0 Standard Architecture and Cluster Architecture all support the sadd, scard, sdiff, sdiffstore, sinter, sinterstore, sismember, smove, spop, srandmember, srem, sscan, sunion, sunionstore, smismember, and smembers commands.
- Cluster Architecture in cross-slot scenarios does not support the sdiff, sdiffstore, sinter, sinterstore, smove, sunion, or sunionstore commands.
- Redis 7.0 version and ValKey 8.0 support: sintercard.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve scenarios of cross-slot access.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support in Cluster Architecture
sadd	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
scard	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
sdiff	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sdiffstore	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sinter	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sinterstore	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sismember	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
smembers	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
smove	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
spop	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
srandmember	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
srem	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
sscan	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
sunion	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
sunionstore	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
smismember	✓	✓	✓	✓	x	x	x	x	x	-
sintercard	✓	✓	x	x	x	x	x	x	x	-

Sorted Sets Group

Last updated: 2026-03-17 18:10:58

- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the `zadd`, `zcard`, `zcount`, `zincrby`, `zinterstore`, `zlexcount`, `zrange`, `zrangebylex`, `zrangebyscore`, `zrank`, `zrem`, `zremrangebylex`, `zremrangebyrank`, `zremrangebyscore`, `zrevrange`, `zrevrangebylex`, `zrevrangebyscore`, `zscore`, `zrevrank`, `zscan`, and `zunionstore` commands.
- Cluster Architecture does not support the `zinterstore` or `zunionstore` commands in scenarios involving cross-slot.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve scenarios involving cross-slot access.

Com mand	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Support for Cross-Slot
zadd	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zcard	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zcount	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zincrby	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zinterstore	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
zlexcount	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrange	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrangebylex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrangebyscore	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrank	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrem	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zremrangebylex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zremrangebyrank	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zremrangebyscore	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrevrange	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrevrangebylex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrevrangebyscore	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zscore	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zrevrank	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zscan	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
zunion	✓	✓	✓	✓	✓	✓	✓	✓	✓	x

store	-	-	-	-	-	-	-	-	-	-
zpop max	✓	✓	✓	✓	✓	✓	x	x	x	-
zpop min	✓	✓	✓	✓	✓	✓	x	x	x	-
bzpop max	✓	✓	✓	✓	✓	✓	x	x	x	-
bzpop min	✓	✓	✓	✓	✓	✓	x	x	x	-
zdiff	✓	✓	✓	✓	x	x	x	x	x	-
zdiffstore	✓	✓	✓	✓	x	x	x	x	x	-
zinter	x	x	x	x	x	x	x	x	x	-
zrandmember	✓	✓	✓	✓	x	x	x	x	x	-
zrange store	✓	✓	✓	✓	x	x	x	x	x	-
zunion	✓	✓	✓	✓	x	x	x	x	x	-
zmscore	✓	✓	✓	✓	x	x	x	x	x	-
zpop	✓	✓	x	x	x	x	x	x	x	-
bzpop	✓	✓	x	x	x	x	x	x	x	-
zintercard	✓	✓	x	x	x	x	x	x	x	-

Strings Group

Last updated: 2026-03-17 18:10:59

- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the append, bitcount, bitop, bitpos, decr, decrby, get, getbit, getrange, getset, incr, incrby, incrbyfloat, mget, mset, msetnx, psetex, setex, set, setbit, setnx, setrange, and strlen commands.
- Redis version 2.8 Standard Architecture does not support bitfield, while Redis versions 4.0, 5.0, and 6.2 all support it.
- Redis 4.0 and 5.0 do not support the bitfield_ro, getdel, and stralgo commands.
- 6.2, 7.0, and ValKey 8.0 version support bitfield_ro, getdel, stralgo, getex.
- Cluster Architecture in cross-slot scenarios supports the mget and mset commands, but does not support the bitop and msetnx commands.
- Redis version 7.0 and ValKey 8.0 support the lcs command, as detailed below.

Command Family	Command	New Description
string family	SET key value [NX XX] [GET] [EX seconds PX milliseconds EXAT unix-time-seconds PXAT unix-time-milliseconds KEEPTTL]	<p>New parameters added: NX and GET.</p> <ul style="list-style-type: none"> • NX: Not exists, sets the key-value pair only when the key does not exist. • GET: Sets the new value and returns the old value.
	BITPOS key bit [start [end [BYTE BIT]]]	<p>New parameters added: BYTE and BIT. Specifies the unit of offset. BYTE: indicates byte offset; BIT: indicates bit offset. Defaults to BYTE.</p>
	LCS key1 key2 [LEN] [IDX] [MINMATCHLEN min-match-len] [WITHMATCHLEN]	<ul style="list-style-type: none"> • New commands added. • Returns the longest common subsequence of two keys.

Specific support information for each version, see the table below. ✓ indicates support, x indicates no support, – indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Cross-Slot Support
append	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
bitcount	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
bitop	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
bitpos	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
decr	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
decrby	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
get	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
getbit	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
getrange	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
getset	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
incr	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
incrby	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
incrbyfloat	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
mget	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
mset	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
msetnx	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
psetex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
setex	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
set	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
setbit	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
setnx	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
setrange	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
strlen	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
bitfield	✓	✓	✓	✓	✓	✓	✓	✓	x	-
bitfield_ro	✓	✓	✓	✓	x	x	x	x	x	-
stralg	✓	✓	✓	✓	x	x	x	x	x	-

o										
getdel	✓	✓	✓	✓	x	x	x	x	x	-
getex	✓	✓	✓	✓	x	x	x	x	x	-
lcs	✓	✓	x	x	x	x	x	x	x	-

Transactions

Last updated: 2026-03-17 18:10:59

Redis 2.8 Standard Architecture, Redis 4.0, 5.0, 6.2, 7.0, and their ValKey 8.0 Standard Architecture and Cluster Architecture all support the discard, exec, multi, unwatch, and watch commands. For specific information, see the table below. ✓ indicates support, x indicates no support, and – indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support for Cluster Architecture
discard	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
exec	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
multi	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
unwatch	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
watch	✓	✓	✓	✓	✓	✓	✓	✓	✓	–

Hyperloglog Group

Last updated: 2026-03-17 18:10:59

Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the pfadd, pfcoun, and pfmerge commands. For details, see the table below. ✓ indicates support, x indicates no support, and – indicates that the command does not involve cross-slot access.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support in Cluster Architecture
pfadd	✓	✓	✓	✓	✓	✓	✓	✓	✓	–
pfcoun	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
pfmerge	✓	✓	✓	✓	✓	✓	✓	✓	✓	x

scripting and functions Group

Last updated: 2026-03-17 18:10:59

Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the `eval`, `evalsha`, `script exists`, `script flush`, `script load`, and `script kill` commands, but do not support the `script debug` command. The `eval`, `evalsha`, and `script exists` commands are not supported in cross-slot scenarios for the Cluster Architecture. Version 7.0 introduces function-related commands, as shown in the following table.

Command Family	Command	New Description
function (added in Redis 7.0)	FUNCTION DELETE library-name	<ul style="list-style-type: none"> Added the function feature. Delete a lib and its functions.
	FUNCTION DUMP	Export all functions.
	FUNCTION FLUSH [ASYNC SYNC]	Delete all libs and their functions.
	FUNCTION KILL	Kill the currently executing function.
	FUNCTION LIST [LIBRARYNAME library-name-pattern] [WITHCODE]	Return information about the lib and functions.
	FUNCTION LOAD [REPLACE] function-code	Load/Replace functions.

	FUNCTION RESTORE serialized-value [FLUSH APPEND REPLACE]	is used to restore snapshots previously saved using the SAVE or BGSAVE commands. <ul style="list-style-type: none"> ● FLUSH: Before a snapshot is restored, all data in the current database will be cleared. ● APPEND: Before a snapshot is restored, the data in the current database will be appended to the snapshot. ● REPLACE: Before a snapshot is restored, the data in the current database will be completely replaced by the data in the snapshot.
	FUNCTION STATS	Return the status of the currently executing function.
	FCALL function numkeys [key [key ...]] [arg [arg ...]]	Invoke a function.
	FCALL_RO function numkeys [key [key ...]] [arg [arg ...]]	FCALL Readonly edition.
Scripting Family	EVAL_RO script numkeys [key [key ...]] [arg [arg ...]]	New command to run lua scripts in read-only replicas.
	EVALSHA_RO sha1 numkeys [key [key ...]] [arg [arg ...]]	New command to run lua scripts in read-only replicas.

Specific support details for each version, see the table below. ✓ indicates supported, x indicates not supported, – indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Cross-Slot Support
eval	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
eval_ro	✓	✓	x	x	x	x	x	x	x	-
evalsha	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
evalsha_ro	✓	✓	x	x	x	x	x	x	x	-
script debug	x	x	x	x	x	x	x	x	x	-
script exists	✓	✓	✓	✓	✓	✓	✓	✓	✓	x
script flush	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
script load	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
script kill	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
function delete	✓	✓	x	x	x	x	x	x	x	-
function kill	✓	✓	x	x	x	x	x	x	x	-
function dump	✓	✓	x	x	x	x	x	x	x	-
function flush	✓	✓	x	x	x	x	x	x	x	-
function list	✓	✓	x	x	x	x	x	x	x	-
function load	✓	✓	x	x	x	x	x	x	x	-
function restore	✓	✓	x	x	x	x	x	x	x	-
function status	✓	✓	x	x	x	x	x	x	x	-
fcall	✓	✓	x	x	x	x	x	x	x	-
fcall_ro	✓	✓	x	x	x	x	x	x	x	-

Geo Group

Last updated: 2026-03-17 18:10:59

Redis 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the geoadd, geohash, geopos, geodist, georadius, and georadiusbymember commands. Only versions 6.2 and 7.0 support the geosearch and geosearchstore commands. The Standard Architecture of Redis 2.8 does not support any of these commands. For details, see the table below. ✓ indicates support, x indicates no support, and – indicates that the command does not involve cross-slot access scenarios.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support in Cluster Architecture
geoadd	✓	✓	✓	✓	✓	✓	✓	✓	x	–
geohash	✓	✓	✓	✓	✓	✓	✓	✓	x	–
geopos	✓	✓	✓	✓	✓	✓	✓	✓	x	–
geodist	✓	✓	✓	✓	✓	✓	✓	✓	x	–
georadius	✓	✓	✓	✓	✓	✓	✓	✓	x	–
georadiusbymember	✓	✓	✓	✓	✓	✓	✓	✓	x	–
geosearch	✓	✓	✓	✓	x	x	x	x	x	–
geosearchstore	✓	✓	✓	✓	x	x	x	x	x	–

Server Group

Last updated: 2026-03-17 18:10:59

- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) support the client list, client getname, client setname, command count, command info, config rewrite, config resetstat, role, lastsave, command, dbsize, info, time, config get, flushdb, and flushall commands.
- Redis versions 2.8 (Standard Architecture), 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (both Standard Architecture and Cluster Architecture) do not support commands such as bgrewriteaof, bgsave, client kill, sync, psync, client pause, client reply, slaveof, config set, debug object, debug segfault, save, shutdown, module, acl cat, acl deluser, acl genpass, acl getuser, acl help, acl list, acl load, acl log, acl save, acl setuser, acl users, acl whoami, and failover.
- Redis versions 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 (Standard Architecture) support the MEMORY and slowlog commands, while the cluster architecture does not support them.
- Redis versions 4.0, 5.0, 6.2, 7.0, and ValKey 8.0 cluster architecture support the cluster keyslot, cluster nodes, cluster getkeysinslot, cluster slots, cluster info, and cluster countkeysinslot commands, while Standard Architecture does not support them. Other cluster commands are not supported.
- The lolwut command is supported only in Redis 5.0 and later versions, both in Standard Architecture and Cluster Architecture.
- The monitor command is not supported in Redis version 6.2, but it is supported in versions prior to 6.2.
- Redis version 7.0 and ValKey 8.0 support the following commands: command getkeysandflags, command docs, command list.

Command Family	Command	New Description
server Family	CONFIG SET/GET	New parameters added for atomically handling multiple configuration setting and getting operations.
	Quit	Marked as deprecated, clients can directly close the TCP connection.
	SHUTDOWN [NOSAVE SAVE] [NOW] [FORCE] [ABORT]	Added NOW FORCE ABORT. <ul style="list-style-type: none"> • NOW: Close immediately without waiting for any ongoing commands to complete. • FORCE: Force close the Redis server even if commands are being executed. • ABORT: Terminate the currently executing command without executing any other

	commands.
COMMAND GETKEYSANDFLAGS command [arg [arg ...]]	New commands added to obtain the keys and flags of commands.
COMMAND DOCS [command-name [command-name ...]]	Added a new command that returns the documentation information of the command, including historical changes and other records.
COMMAND LIST [FILTERBY <MODULE module-name ACLCAT category PATTERN pattern>]	Added command that returns the list of Redis commands.
COMMAND INFO	Update command that returns information about Redis commands.
INFO [section [section ...]]	Added parameters that support obtaining multiple sections simultaneously.
XGROUP CREATE key group <id \$> [MKSTREAM] [ENTRIESREAD entries-read]	Added ENTRIESREAD to configure and enable the entries-read and lag features.
XGROUP SETID key group <id \$> [ENTRIESREAD entries-read]	Added ENTRIESREAD to configure and enable the entries-read and lag features.

For specific support information of each edition, see the table below. ✓ indicates support, x indicates no support, and – indicates that cross-slot access is not applicable for this command.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cluster Architecture Cross-Slot Support
bgrewriteof	x	x	x	x	x	x	x	x	x	-
bgsave	x	x	x	x	x	x	x	x	x	-
client kill	x	x	x	x	x	x	x	x	x	-
sync	x	x	x	x	x	x	x	x	x	-
psync	x	x	x	x	x	x	x	x	x	-
client list	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
client getname	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
client pause	x	x	x	x	x	x	x	x	x	-
client reply	x	x	x	x	x	x	x	x	x	-
client setname	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
command count	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
command getkeys	✓	✓	✓	✓	✓	✓	✓	✓	x	-
command info	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
command getkeysand flags	✓	✓	x	x	x	x	x	x	x	-
command docs	✓	✓	x	x	x	x	x	x	x	-
command list	✓	✓	x	x	x	x	x	x	x	-
slaveof	x	x	x	x	x	x	x	x	x	-
config rewrite	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
config set	x	x	x	x	x	x	x	x	x	-
config resetstat	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
debug object	x	x	x	x	x	x	x	x	x	-
debug segfault	x	x	x	x	x	x	x	x	x	-
role	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
save	x	x	x	x	x	x	x	x	x	-
lastsave	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
shutdown	x	x	x	x	x	x	x	x	x	-
-----	✓	..	✓	..	✓	

memory	x	x	✓	x	✓	x	✓	x	x	-
command	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
dbsize	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
info	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
time	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
config get	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
monitor	x	x	x	x	✓	✓	✓	✓	✓	-
flushdb	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
flushall	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
slowlog	✓	x	✓	x	✓	x	✓	x	✓	-
cluster keyslot	x	✓	x	✓	x	✓	x	✓	x	-
cluster nodes	x	✓	x	✓	x	✓	x	✓	x	-
cluster getkeysinslot	x	✓	x	✓	x	✓	x	✓	x	-
cluster slots	x	✓	x	✓	x	✓	x	✓	x	-
cluster info	x	✓	x	✓	x	✓	x	✓	x	-
cluster countkeysinslot	x	✓	x	✓	x	✓	x	✓	x	-
cluster shards	✓	✓	x	x	x	x	x	x	x	-
cluster other	x	x	x	x	x	x	x	x	x	-
module	x	x	x	x	x	x	x	x	x	-
lolwut	✓	✓	✓	✓	✓	✓	x	x	x	-
acl cat	x	x	x	x	x	x	x	x	x	-
acl deluser	x	x	x	x	x	x	x	x	x	-
acl genpass	x	x	x	x	x	x	x	x	x	-
acl help	x	x	x	x	x	x	x	x	x	-
acl list	x	x	x	x	x	x	x	x	x	-
acl load	x	x	x	x	x	x	x	x	x	-
acl log	x	x	x	x	x	x	x	x	x	-
acl save	x	x	x	x	x	x	x	x	x	-
acl setuser	x	x	x	x	x	x	x	x	x	-
acl users	x	x	x	x	x	x	x	x	x	-
acl whoami	x	x	x	x	x	x	x	x	x	-
failover	x	x	x	x	x	x	x	x	x	-
slowlog help	x	x	x	x	x	x	x	x	x	-

Stream Group

Last updated: 2026-03-17 18:10:59

- Redis 5.0, 6.2, 7.0, and ValKey 8.0 Standard Architecture and Cluster Architecture all support the xinfo, xadd, xtrim, xdel, xrange, xrevrange, xlen, xread, xgroup, xreadgroup, xack, xclaim, and xpending commands. These commands are not supported in Redis 2.8 Standard Architecture, 4.0 Standard Architecture, or Cluster Architecture. Only versions 6.2 and 7.0 support the xautoclaim command.
- Cluster Architecture does not support the xread or xreadgroup commands in cross-slot scenarios.

For specific information, see the table below. "✓" indicates support, "x" indicates no support, and "-" indicates that the command does not involve scenarios for cross-slot access.

Command	Redis 7.0/ValKey 8.0 Standard Architecture	Redis 7.0/ValKey 8.0 Cluster Architecture	Redis 6.2 Standard Architecture	Redis 6.2 Cluster Architecture	Redis 5.0 Standard Architecture	Redis 5.0 Cluster Architecture	Redis 4.0 Standard Architecture	Redis 4.0 Cluster Architecture	Redis 2.8 Standard Architecture	Cross-Slot Support in Cluster Architecture
xinfo	✓	✓	✓	✓	✓	✓	x	x	x	-
xadd	✓	✓	✓	✓	✓	✓	x	x	x	-
xtrim	✓	✓	✓	✓	✓	✓	x	x	x	-
xdel	✓	✓	✓	✓	✓	✓	x	x	x	-
xrange	✓	✓	✓	✓	✓	✓	x	x	x	-
xrevrange	✓	✓	✓	✓	✓	✓	x	x	x	-
xlen	✓	✓	✓	✓	✓	✓	x	x	x	-
xread	✓	✓	✓	✓	✓	✓	x	x	x	x
xgroup	✓	✓	✓	✓	✓	✓	x	x	x	-
xreadgroup	✓	✓	✓	✓	✓	✓	x	x	x	x
xack	✓	✓	✓	✓	✓	✓	x	x	x	-
xclaim	✓	✓	✓	✓	✓	✓	x	x	x	-
xpending	✓	✓	✓	✓	✓	✓	x	x	x	-
xautoclaim	✓	✓	✓	✓	x	x	x	x	x	-

Differences in Redis Major Version Command Usage

Last updated: 2026-01-28 10:07:37

Command/Feature	Cloud Redis 5.0	Cloud Redis 6.2	Cloud Redis 7.0	Description
RESP3	–	–	–	Currently not supported. Assess whether your business framework uses <code>HELLO 3 auth username password</code> for authentication, and whether the RESP3 protocol is used. If so, downgrade to RESP2.
ACL	–	–	–	The access control list (ACL) feature is not supported at present; confirm whether your business relies on this feature. This limitation affects all command APIs related to ACL users, including <code>acl</code> , <code>auth</code> , <code>migrate auth</code> , and <code>client kill user</code> .
Module	–	–	–	Modules are not supported in the current version. Confirm whether your business enables this feature.
client-side-caching	–	–	–	Currently not supported. Confirm whether your business relies on client-side caching optimization.
pub/sub	–	–	–	TencentDB for Redis® Cluster Edition has enhanced the <code>pubsub</code> command. For details, see Differences Between the Proxy Architecture and Direct Connection Mode .
Cluster multi-DB	Multi-DB is supported in cluster mode.			However, it is not recommended to use or rely on this feature. Commands such as <code>select</code> and <code>swapdb</code> are enabled. Some clients may determine cluster mode based on the success of <code>select</code> , which can cause misjudgments.
lua readonly table	Defining non-local functions is not allowed.			To address CVE security vulnerabilities, the community has enabled read-only table protection in the Lua sandbox, which prohibits

redefining global (non-local) functions. Otherwise, execution will fail with the error: ERR Attempt to modify a readonly table ... Perform a full scan of all Lua calls, such as eval, script, and function, and explicitly change all custom functions to use local definitions: `eval "local function xxx () return 'hello' end" 0` .

Note:

- This change is a breaking change, but it has been released by the community as a minor version security patch. Differences may exist across cloud providers and major/minor versions (such as rollback or lack of backporting).
- Do not rely on legacy usage. Businesses should comply with specifications to avoid production failures.
- For more information, see [Lua readonly tables \(CVE-2022-24736, CVE-2022-24735\)](#) .

<p>lua print</p>	<p>lua print is allowed.</p>	<p>lua print is not allowed .</p>	<p>To fix CVE security vulnerabilities, the print function in the Lua sandbox has been removed (breaking change). Continuing to use it will cause an error: <code>ERR ... nonexistent global variable 'print' ...</code> Troubleshoot all eval and script Lua scripts, and delete or replace all print(...) calls. For logging, use the official API: <code>redis.log(redis.LOG_NOTICE, 'hello')</code> .</p>
<p>script load</p>	<ul style="list-style-type: none"> • Scripts loaded with script load are compiled and cached, and the command (including the script content) is disseminated as a write command to all secondary nodes to ensure identical script caches. 	<p>Starting from Redis 7.0 (community edition), Lua scripts are no longer persisted. Cloud Redis 7.0 retains Lua persistence.</p>	

	<ul style="list-style-type: none"> • If AOF persistence is enabled, script load is written to AOF in the same format as received. 	
<p>script flush</p>	<ul style="list-style-type: none"> • The script flush command is disseminated to both primary and secondary nodes and written to the AOF file. When the primary node clears its local script cache, it synchronizes the flush command to the secondary nodes, ensuring that their caches are cleared as well and preventing inconsistencies between the primary and secondary script caches. • If AOF persistence is enabled, script flush is also written to AOF. 	
<p>Lua script persistence</p>	<p>In primary–secondary replication, RDB snapshots also store Lua scripts.</p>	
<p>Lua EVAL script eviction</p>	<p>All the latest minor versions support Lua script eviction.</p> <div data-bbox="287 1227 805 1899" style="border: 1px solid #00aaff; padding: 10px; margin-top: 10px;"> <p>Note:</p> <p>To prevent abuse of EVAL by embedding parameters directly into the script body, Redis introduced an LRU eviction mechanism with a limit of 500 entries. This eviction applies only to scripts generated by EVAL. When the limit is exceeded, the least recently used scripts are automatically evicted. Scripts loaded with SCRIPT LOAD are not affected.</p> </div>	<p>No changes are required for your business. This update only backports the behavior from higher community versions to ensure consistency with the latest official release.</p> <div data-bbox="874 1312 1481 2078" style="border: 1px solid #00aaff; padding: 10px; margin-top: 10px;"> <p>Note:</p> <p>In typical scenarios, scripts are fully delivered through EVAL each time. Even if they are evicted by the LRU mechanism, it does not affect your business. However, if users adopt a nonstandard approach—first loading the script onto the server using EVAL and manually computing the SHA, then calling EVALSHA—when a single transaction involves more than 500 scripts, the least recently used script may be evicted. This can cause subsequent EVALSHA calls to return</p> </div>

		<p>a NOSCRIPT error and lead to transaction failure.</p>
<p>Lua memory abuse limit</p>	<p>All latest minor versions across all cloud major versions support Lua memory limits.</p> <div data-bbox="288 488 805 1406" style="border: 1px solid #00a0e3; padding: 10px;"> <p>Note:</p> <p>In the community edition, Lua does not verify maxmemory when executing write commands. This allows malicious scripts (such as infinite loops with SET, <code> EVAL "for i=1,1000000000 do redis.call('SET', 'key-:' .. i, 'value-:' .. i) end" 0)</code> to bypass the memory upper limit. To prevent machine OOM in cloud environments, memory usage is monitored in real time during Lua execution. Once the limit is exceeded, an OOM error is returned immediately.</p> </div>	<p>Normal business workloads are not affected as long as the instance has sufficient maxmemory. The cloud-side also provides the extra-maxmemory parameter, which can reserve additional memory space for an instance when necessary.</p> <div data-bbox="874 891 1481 1131" style="border: 1px solid #00a0e3; padding: 10px;"> <p>Note:</p> <p>If an OOM is triggered, the Lua script will fail midway, and atomicity cannot be guaranteed.</p> </div>
<p>Dictionary rehash optimization</p>	<p>All latest minor versions across all cloud major versions support dictionary rehash optimization.</p> <div data-bbox="288 1615 805 2074" style="border: 1px solid #00a0e3; padding: 10px;"> <p>Note:</p> <p>In the community edition, dictionary implementation uses an incremental rehash mechanism consisting of both active and passive rehash. Passive rehash occurs during the processing of each user request, which triggers</p> </div>	<div data-bbox="874 1532 1481 2074" style="border: 1px solid #00a0e3; padding: 10px;"> <p>Note:</p> <p>Disabling passive rehash slows down the rehash process and may slightly increase the probability of hash collisions. Temporary memory used during rehash may also remain longer in monitoring metrics compared with the community edition. However, user requests generally avoid additional performance jitters from passive</p> </div>

	<p>additional rehash operations. This results in extra memory accesses, increasing request latency and reducing node throughput. To improve performance, the cloud edition disables passive rehash, retaining only active rehash.</p>	<p>rehash, which benefits most use cases. Only in certain rare extreme cases, such as very high-frequency write workloads, will performance regress to levels close to the community edition. To address this, the cloud edition provides the passive-rehash-enabled configuration option for flexible control over enabling or disabling passive rehash.</p>
<p>Replica primary – secondary disconnection redirection</p>	<p>All the latest minor versions across all cloud major versions support replica primary–secondary disconnection redirection.</p> <div data-bbox="288 958 807 1491" style="border: 1px solid #00aaff; padding: 10px;"> <p>Note: When a replica disconnects from the primary node, before failover completes, read requests sent to the replica are redirected to the primary node. This prevents serving stale data while the replica is in read-only mode due to the disconnection.</p> </div>	<p>No special processing is required from your business.</p>
<p>info/client info/client list</p>	<p>Field support across different versions may not be fully compatible with the community edition. The proxy layer will block sensitive fields as needed. For details, see Differences Between the Proxy Architecture and Direct Connection Mode.</p>	<p>Check whether your business relies on the client command. If not required, avoid executing client list to prevent slow queries and excessive node memory usage.</p>
<p>cluster slots</p>	<p>Cluster Edition hides or converts certain sensitive information in cluster slots and other commands.</p>	<p>For details, see Differences Between the Proxy Architecture and Direct Connection Mode.</p>

<p>Commands such as auth/migrate auth/client kill user/ACL setuser and others</p>	<p>No ACL component.</p>	<p>ACL-related APIs are not supported.</p>	<p>—</p>
<p>command/command info</p>	<p>The command command provides metadata about commands to client libraries for routing decisions, redirection, and other operations.</p>		<p>No special processing is required from your business.</p>
<p>geo* store</p>	<p>The following issues were fixed in Redis 6.2 and 7.0: For geo commands (such as georadius, georadiusbymember, and geosearchstore) that support the store option, when the source key is empty, the behavior was changed from "return an empty array without deleting the target key" to "delete the target key and return 0."</p>		 <p>To reproduce</p> <pre>GEOADD destination 0 0 member (integer) 1 GEOSEARCHSTORE destination source FROMLONLAT 0 0 BYRADIUS 1 km (empty array) ZCARD destination (integer) 1</pre> <p>Expected behavior</p> <pre>GEOADD destination 0 0 member (integer) 1 GEOSEARCHSTORE destination source FROMLONLAT 0 0 BYRADIUS 1 km (integer) 0 ZCARD destination (integer) 0</pre>
<p>zranges store</p>	<p>Fixed an issue where the ZRANGESTORE command could crash Redis when the configuration item zset-max-ziplist-entries was set to 0.</p>		<p>—</p>
<p>lpop/rpop count</p>	<p>The return behavior of LPOP/RPOP was unified in versions 6.2.7 and 7.0.0:</p> <ol style="list-style-type: none"> 1. When the key exists and count = 0, an empty array [] is returned (previously returned nil). 2. When the key does not exist and the count parameter is specified, an empty array [] with length -1 is returned (previously returned nil). 		 <pre>127.0.0.1:6379> lpush mylist a b c 127.0.0.1:6379> lpush mylist a b c (integer) 3 (integer) 3 127.0.0.1:6379> lpop mylist 0 127.0.0.1:6379> lpop mylist 0 (nil) (empty array) 127.0.0.1:6379> []</pre>

Differences Between the Proxy Architecture and Direct Connection Mode

Last updated: 2026-03-17 18:10:59

Command	Redis 5.0	Redis 6.2	Redis 7.0	Self-Built Direct Connection Mode	Suggestion
del					Normal use.
unlink					Normal use.
exists					Normal use.
mset					Normal use.
mget	Cluster Edition supports cross-slot and cross-node command operations. The Proxy layer automatically completes key distribution and result aggregation.			Cross-slot and cross-node operations are not supported.	Normal use.
zunionstore					<div style="border: 1px solid #00a88f; padding: 5px; margin-bottom: 5px;"> <p>Note: Proxy will return an error if the verification of the mget command does not match expectations. <code>-ERR invalid return data</code>.</p> </div> <p>Since cross-slot operations require the Proxy to perform additional processing for command distribution and result aggregation, it is recommended to follow Redis community best practices to ensure related keys are placed in the same slot for better performance.</p>
randomkey	The randomkey command in Cluster Edition is randomly forwarded to			It is executed	Recommended to use only when random keys are explicitly required

	different backend nodes.	ted only on the currently connected node.	from the current shard; otherwise, use SCAN or other alternatives.
keys	<ul style="list-style-type: none"> The keys command in Cluster Edition is executed in parallel on all master nodes by default. The Proxy layer aggregates the results from each shard and returns them to the client. Cluster Edition supports the syntax <code>keys [@shard_id]</code> to specify shard execution. In this case, only the execution result of the target shard is returned, without cross-node aggregation. The keys command in Standard Edition is directly forwarded to the only master node, with no shard aggregation, and its response behavior is consistent with native Redis. 	It is executed only on the currently connected node.	<p>Select the execution method of the keys command based on business requirements: shard-specific execution (returns data from a single shard) or all-shard execution (returns aggregated data). Execution methods are as follows:</p> <ul style="list-style-type: none"> KEYS {hash_tag}pattern KEYS pattern NODEID
scan	<ul style="list-style-type: none"> The scan command in Cluster Edition polls all master nodes by default. The scan command in Standard Edition is directly forwarded to the master node. Both Cluster and Standard Editions support execution on specified backend nodes (such as <code>SCAN 0 @shard2</code>). 	It is executed only on the currently connected node.	<p>Select the execution method of the scan command based on business requirements: shard-specific execution (return data from a single shard) or all-shard execution (return aggregated data). Execution methods are as follows:</p> <ul style="list-style-type: none"> SCAN cursor [MATCH {hash_tag}pattern] [COUNT count] SCAN cursor [MATCH pattern] [COUNT count] NODEID
dbsize	<ul style="list-style-type: none"> The dbsize command in Cluster Edition is executed on all master nodes by default. The Proxy layer 	It is executed	Select the execution method of the dbsize command based on business requirements: shard-specific

	<p>aggregates the results from each shard and returns them to the client.</p> <ul style="list-style-type: none"> Cluster Edition supports the syntax <code>dbsize [@shard_id]</code> to specify shard execution. In this case, only the execution result of the target shard is returned, without cross-node aggregation. The <code>dbsize</code> command in Standard Edition is directly forwarded to the only master node, with no shard aggregation, and its response behavior is consistent with native Redis. 	<p>only on the currently connected node.</p>	<p>execution (returns data from a single shard) or all-shard execution (returns aggregated data). Execution methods are as follows:</p> <ul style="list-style-type: none"> <code>DBSIZE</code> <code>DBSIZE NODEID</code>
flushdb	<ul style="list-style-type: none"> The <code>flushdb</code> command in Cluster Edition is executed on all master nodes by default. The Proxy layer aggregates the results from each shard and returns them to the client. Cluster Edition supports the syntax <code>flushdb [@shard_id]</code> to specify shard execution. In this case, only the execution result of the target shard is returned, without cross-node aggregation. Versions 6.2 and 7.0 support the <code>ASYNC</code> and <code>SYNC</code> options, with <code>ASYNC</code> as the default. 	<p>It is executed only on the currently connected node.</p>	<p>Select the execution method of the <code>flushdb</code> command based on business requirements: shard-specific execution (returns data from a single shard) or all-shard execution (returns aggregated data). Command usage is as follows:</p> <ul style="list-style-type: none"> <code>FLUSHDB</code> <code>FLUSHDB SYNC</code> <code>FLUSHDB ASYNC</code> <code>FLUSHDB SYNC NODEID</code> <code>FLUSHDB ASYNC NODEID</code>
flushall	<ul style="list-style-type: none"> The <code>flushall</code> command in Cluster Edition is executed on all master nodes by default. The Proxy layer aggregates the results from each shard and returns them to the client. Cluster Edition supports the syntax <code>flushall [@shard_id]</code> to specify shard execution. 	<p>It is executed only on the currently connected</p>	<p>Select the execution method of the <code>flushall</code> command based on business requirements: shard-specific execution (returns data from a single shard) or all-shard execution (returns aggregated data). Command usage is as follows:</p> <ul style="list-style-type: none"> <code>FLUSHALL</code> <code>FLUSHALL SYNC</code>

	<p>In this case, only the execution result of the target shard is returned, without cross-node aggregation.</p> <ul style="list-style-type: none"> • Versions 6.2 and 7.0 support the ASYNC and SYNC options, with ASYNC as the default. 	<p>ected node.</p>	<ul style="list-style-type: none"> • FLUSHALL ASYNC • FLUSHALL SYNC NODEID • FLUSHALL ASYNC NODEID
subscribe	<ul style="list-style-type: none"> • Commands in Cluster Edition are forwarded to all master nodes by default. • Commands in Standard Edition are forwarded to the master node by default. 	<p>It is executed only on the currently connected node.</p>	Normal use.
unsubscribe			Normal use.
psubscribe			Normal use.
punsubscribe			Normal use.
publish	<ul style="list-style-type: none"> • The publish command in Cluster Edition is routed to the corresponding master node for broadcasting based on the slot value calculated from the channel name. • The publish command in Standard Edition is forwarded to the master node by default. 	<p>It is executed only on the currently connected node.</p>	Normal use.
pubsub	<ul style="list-style-type: none"> • Supports CHANNELS, NUMPAT, and NUMSUB options. • The pubsub command in Cluster Edition is randomly forwarded to different backend master nodes by default, but the returned results are consistent. • The pubsub command in Standard Edition is forwarded to the master node by default. 	<p>Executed only on the currently connected node, and different</p>	<p>TencentDB for Redis® Cluster Edition enhances the pubsub command. All nodes return a globally consistent channel list, which differs from the open-source version, where nodes return inconsistent results. It is recommended that business code avoid relying on shard-local status and instead process subscriptions on a cluster-wide basis.</p>

				nodes may return different results.	
cluster	<ul style="list-style-type: none"> Cluster Edition supports the following commands: cluster keyslot, cluster slots, cluster nodes, cluster info, cluster countkeysinslot, and cluster getkeysinslot. The cluster slots, cluster nodes, and cluster info commands in Cluster Edition are randomly forwarded to backend nodes by default. The cluster info command additionally supports execution on a specified backend node. The cluster keyslot, cluster countkeysinslot, cluster keyslot, and cluster getkeysinslot commands in Cluster Edition are forwarded to the corresponding node based on the specified slot. Cluster Edition hides or converts certain sensitive information in cluster slots, cluster nodes, and other commands. 			It is executed only on the currently connected node.	Assess the forwarding mode of cluster commands based on your business scenarios.
asking	The command performs no actual operation in both Cluster Edition and Standard Edition.			Execution results in redirection.	Proxy processes ASK errors automatically; users do not need to intervene.
function	-	-	<ul style="list-style-type: none"> Cluster Edition supports 	It is executed	It is recommended to assess whether cloud-based architecture differs from

the following commands: function stats, function dump, function list. These commands are randomly forwarded to backend nodes by default.

- The function delete, function flush, function kill, function load, and function restore commands are executed synchronously on all master nodes by default, and the client receives a response only after all nodes return consistent results.

only on the currently connected node.

your business scenarios. If differences exist, modify your code accordingly.

script	<ul style="list-style-type: none"> ● In both Cluster Edition and Standard Edition, the script debug command is not supported. ● The script exists, script flush, script kill, and script load commands in Cluster Edition are executed synchronously on all master nodes by default, and the client receives a response only after all nodes return consistent results. ● The script command in Standard Edition is forwarded to the master node by default. 	It is executed only on the currently connected node.	Check whether your existing code logic differs from Tencent Cloud's default all-node execution logic. If inconsistencies exist, refactor your code according to cloud specifications.	
multi exec discard	<ul style="list-style-type: none"> ● The backend node locked by a transaction or watch is determined by the hash slot of the first operated key (with watch taking precedence over the first command after multi). This ensures atomic execution of transactions or watch operations within a single node. ● Commands in Standard Edition are forwarded to the master node by default. 	It is executed only on the currently connected node.	It is recommended to assess whether the usage of transactions and the watch command differs from that before migrating to the cloud.	
watch	After watch is executed, the connection is locked to the node where the monitored key resides. All subsequent commands are forwarded through this connection.	It is executed only on the currently connected node.	It is recommended that key operations between watch and unwatch commands remain within the same slot.	
expire at	-	In both Cluster Edition and Standard Edition, the	It is executed only on	Assess whether your business requires the following options: NX XX GT LT.

		expireat command does not support any of the following options: NX XX GT LT.	the currently connected node, but supports the following options: NX XX GT LT.	
pexpire	-	In both Cluster Edition and Standard Edition, the pexpire command does not support any of the following options: NX XX GT LT.		
pexpireat	-	In both Cluster Edition and Standard Edition, the pexpireat command does not support any of the following options: NX XX GT LT.		
client	<ul style="list-style-type: none"> Both Cluster Edition and Standard Edition fully support client list, client setname, client getname, client id, client kill, and client unblock commands at the Proxy layer. The implementation of the client list relies on a dedicated subscription connection: Each Proxy maintains this connection with the master responsible for slot 0. It is used to aggregate 		It is executed only on the currently connected node, retur	Check whether your business relies on the client command. If not required, avoid executing client list to prevent slow queries and excessive node memory usage.

	<p>session information across all Proxies in real time.</p> <ul style="list-style-type: none"> • The output of the client list only includes id, addr, fd, name, cmd, age, idle, and proxy fields, with id always being negative. The format differs from the open-source edition. • The client kill command can directly terminate sessions on other Proxies. 	<p>ning client list information according to the corresponding Redis version.</p>	
monitor	<p>The monitor command only monitors commands executed on the currently connected Proxy.</p>	<p>It is executed only on the currently connected node.</p>	<p>Ensure that you clearly define whether the monitoring target is a single Proxy or metrics of a specific backend shard.</p>
imonitor	<ul style="list-style-type: none"> • The imonitor command in Cluster Edition supports monitoring commands executed on specified backend nodes. • The imonitor command in Standard Edition is forwarded only to the master node. 	<p>This command is not available.</p>	<p>Use as follows:</p> <ul style="list-style-type: none"> • Cluster Edition: IMONITOR NODEID • Standard Edition: IMONITOR
time	<ul style="list-style-type: none"> • The time command in Cluster Edition is randomly forwarded to different backend nodes by default. • The time command in Standard Edition is forwarded to the master node by default. 	<p>It is executed only on the currently conn</p>	<p>Assess whether your business depends on the time command.</p>

		ected node.	
wait	<ul style="list-style-type: none"> The wait command in Cluster Edition is forwarded based on the backend master node that processed the last write command. The wait command in Standard Edition is forwarded to the master node by default. 	It is executed only on the currently connected node.	Assess the differences between the proxy forwarding logic on the cloud and that in your application code.
command	<ul style="list-style-type: none"> The command command in Cluster Edition is randomly forwarded to different backend nodes by default. The command command in Standard Edition is forwarded to the default master node. 	It is executed only on the currently connected node.	Normal use.
lolwut	<ul style="list-style-type: none"> The lolwut command in Cluster Edition is randomly forwarded to different backend nodes by default. The lolwut command in Standard Edition is forwarded to the master node by default. 	It is executed only on the currently connected node.	Normal use.

echo	<ul style="list-style-type: none"> The echo command in Cluster Edition is randomly forwarded to different backend master nodes by default. Commands in Standard Edition are forwarded to the master node by default. 	It is executed only on the currently connected node.	Normal use.
info	<ul style="list-style-type: none"> The info command in Cluster Edition is randomly forwarded to backend master nodes by default. The info command in Standard Edition is forwarded to the master node by default. The info command also supports specifying a backend node for forwarding. The info command masks and transforms sensitive information in the response. 	It is executed only on the currently connected node.	<p>info command usage is as follows:</p> <ul style="list-style-type: none"> INFO INFO NODEID
ping	<ul style="list-style-type: none"> The ping command in Cluster Edition is randomly forwarded to any backend master node by default. The ping command additionally supports execution on a specified backend node. In both Cluster Edition and Standard Edition, the ping command can be processed directly by the Proxy without being forwarded to any backend node. 	It is executed only on the currently connected node.	<p>Assess whether your business requires forwarding the ping command to backend shards. Command usage is as follows:</p> <ul style="list-style-type: none"> PING PING NODEID
hello	<p>–</p> <ul style="list-style-type: none"> In both Cluster Edition and Standard Edition, the hello command currently supports only version 	It is executed only on	<p>Assess whether your business framework uses <code>HELLO 3 auth username password</code> for authentication, and whether the</p>

		<p>2 and requires the user to execute the Auth command before it can be executed successfully.</p> <ul style="list-style-type: none"> Proxy does not support the RESP3 protocol. Executing hello 3 will return an error. Update your client framework to use the RESP2 auth command for authentication. 	the currently connected node.	RESP3 protocol is used. If so, downgrade to RESP2.
config	<ul style="list-style-type: none"> In both Cluster Edition and Standard Edition, the config set command is intercepted by the Proxy, which immediately returns a "success" response, but the command does not actually take effect. The config get command in Cluster Edition is randomly sent to any backend node by default. By default, the Standard Edition accesses the master node. 		It is executed only on the currently connected node.	It is recommended to view or modify the configuration via the console. For details, see Parameter Configuration .
slowlog	<ul style="list-style-type: none"> The slowlog command in Cluster Edition should specify a backend node for execution; otherwise, an error will occur. The slowlog command in Standard Edition is forwarded to the master node by default. 		It is executed only on the currently connected node.	It is recommended to view complete slow query information via the console. For details, see Querying Redis Slow Logs .
memory	The memory command is not supported in Cluster Edition.		It is executed only	It is recommended to view instance Memory Analysis (Large Key Analysis) via the console.

		on the currently connected node.	
readonly	The readonly command in Cluster Edition is executed on Proxy and returns success, but has no actual effect.	It is executed and takes effect only on the currently connected node.	It is recommended to enable replica read-only mode via the console. For details, see Replica Read-Only .
debug	The debug command supports only the object subcommand.	All debug subcommands are supported.	It is recommended to assess whether your business uses other debug subcommands. If so, code modification will be required.
sentinel	In both Cluster Edition and Standard Edition, the SENTINEL sentinels, SENTINEL get-master-addr-by-name, and SENTINEL slaves commands are supported but not forwarded to backend nodes.	Cluster Edition does not support sentinel mode.	<ul style="list-style-type: none"> For Cluster Edition, these commands can be ignored. For sentinel mode, assess whether the supported sentinel commands on cloud Redis meet your business requirements.

role	<ul style="list-style-type: none"> • The role command in Cluster Edition is randomly forwarded to any backend node by default. • Commands in Standard Edition are forwarded to the master node by default. 	It is executed only on the currently connected node.	It is recommended not to use this command.
lastsave	<ul style="list-style-type: none"> • Commands in Cluster Edition are randomly forwarded to different backend master nodes by default. • Commands in Standard Edition are forwarded to the master node by default. 	It is executed only on the currently connected node.	It is recommended not to use this command.
latency	<ul style="list-style-type: none"> • Not supported in Cluster Edition. • Commands in Standard Edition are forwarded to the default master node. 	It is executed only on the currently connected node.	It is recommended not to use this command.
readwrite	The readwrite command is not supported in Cluster Edition.	It is executed only on the currently connected	It is recommended not to use this command.

		ected node.	
pfselftest	The pfselftest command is not supported in Cluster Edition.	It is executed only on the currently connected node.	It is recommended not to use this command.
post	The post command is not supported in Cluster Edition.	It is executed only on the currently connected node.	It is recommended not to use this command.
host	The host command is not supported in Cluster Edition.	It is executed only on the currently connected node.	It is recommended not to use this command.

More Command Operations (Redis/Valkey Edition)

List of Supported DMC Commands

Last updated: 2026-03-17 18:11:00

DMC (Database Management Center) is a database management tool from Tencent Cloud that allows convenient access to instances, viewing key metrics of instances, running Redis commands, and so on. The Redis commands currently supported by DMC are shown in the table below. Custom commands supported by DMC include: INFO, SLOWLOG, SCAN. For details, see [Custom Command Usage Example](#). In the table below, ✓ indicates supported, x indicates not supported, and – indicates that the command does not involve cross-slot access scenarios.

Command Family	Command	2.8 Standard Architecture	Valkey 8.0/Redis 4.0 and above Standard Architecture)	Valkey 8.0/Redis 4.0 Cluster Architecture	Cluster Architecture Cross Slot Support
Connection Family	echo	✓	✓	✓	–
	ping	✓	✓	Custom	–
	select	✓	✓	✓	–
Hash Family	hdel	✓	✓	✓	–
	hexists	✓	✓	✓	–
	hget	✓	✓	✓	–
	hgetall	✓	✓	✓	–
	hincrby	✓	✓	✓	–
	hincrbyfloat	✓	✓	✓	–
	hkeys	✓	✓	✓	–
	hlen	✓	✓	✓	–
	hmget	✓	✓	✓	–

	hmset	✓	✓	✓	–
	hset	✓	✓	✓	–
	hsetnx	✓	✓	✓	–
	hstrlen	✓	✓	✓	–
	hvals	✓	✓	✓	–
	hscan	✓	✓	✓	–
Keys Family	del	✓	✓	✓	✓
	scan	✓	✓	Custom	–
	exists	✓	✓	✓	✓
	expire	✓	✓	✓	–
	expireat	✓	✓	✓	–
	type	✓	✓	✓	–
	ttl	✓	✓	✓	–
	persist	✓	✓	✓	–
	pexpire	✓	✓	✓	–
	pexpireat	✓	✓	✓	–
	pttl	✓	✓	✓	–
	randomkey	✓	✓	✓	–
	rename	✓	✓	✓	x
	renamenx	✓	✓	✓	x
	touch	✓	✓	✓	–
	restore	✓	✓	✓	–
	unlink	x	✓	✓	x
	move	✓	✓	✓	–
	dump	✓	✓	✓	–

list Family	lindex	✓	✓	✓	–
	linsert	✓	✓	✓	–
	llen	✓	✓	✓	–
	lpop	✓	✓	✓	–
	lpush	✓	✓	✓	–
	lpushx	✓	✓	✓	–
	lrange	✓	✓	✓	–
	lrem	✓	✓	✓	–
	lset	✓	✓	✓	–
	ltrim	✓	✓	✓	–
	rpop	✓	✓	✓	–
	rpoplpush	✓	✓	✓	x
	rpush	✓	✓	✓	–
	rpushx	✓	✓	✓	–
	blpop	✓	✓	✓	x
	brpop	✓	✓	✓	x
	brpoplpush	✓	✓	✓	x
sets Family	sadd	✓	✓	✓	–
	scard	✓	✓	✓	–
	sdiff	✓	✓	✓	x
	sdiffstore	✓	✓	✓	x
	sinter	✓	✓	✓	x
	sinterstore	✓	✓	✓	x
	sismember	✓	✓	✓	–
	smembers	✓	✓	✓	–

	smove	✓	✓	✓	X
	spop	✓	✓	✓	-
	srandmember	✓	✓	✓	-
	srem	✓	✓	✓	-
	sscan	✓	✓	✓	-
	sunion	✓	✓	✓	X
	sunionstore	✓	✓	✓	X
Sorted Sets Family	zadd	✓	✓	✓	-
	zcard	✓	✓	✓	-
	zcount	✓	✓	✓	-
	zincrby	✓	✓	✓	-
	zinterstore	✓	✓	✓	X
	zlexcount	✓	✓	✓	-
	zrange	✓	✓	✓	-
	zrangebylex	✓	✓	✓	-
	zrangebyscore	✓	✓	✓	-
	zrank	✓	✓	✓	-
	zrem	✓	✓	✓	-
	zremrangebylex	✓	✓	✓	-
	zremrangebyrank	✓	✓	✓	-
	zremrangebyscore	✓	✓	✓	-
zrevrange	✓	✓	✓	-	

	zrevrangeby lex	✓	✓	✓	–
	zrevrangeby score	✓	✓	✓	–
	zscore	✓	✓	✓	–
	zrevrank	✓	✓	✓	–
	zscan	✓	✓	✓	–
	zunionstore	✓	✓	✓	X
	zpopmax	X	X	X	–
	zpopmin	X	X	X	–
	bzpopmax	X	X	X	–
	bzpopmin	X	X	X	–
strings Family	append	✓	✓	✓	–
	bitcount	✓	✓	✓	–
	bitop	✓	✓	✓	X
	bitpos	✓	✓	✓	–
	decr	✓	✓	✓	–
	decrby	✓	✓	✓	–
	get	✓	✓	✓	–
	getbit	✓	✓	✓	–
	getrange	✓	✓	✓	–
	getset	✓	✓	✓	–
	incr	✓	✓	✓	–
	incrby	✓	✓	✓	–
	incrbyfloat	✓	✓	✓	–
	mget	✓	✓	✓	✓

	mset	✓	✓	✓	✓
	msetnx	✓	✓	✓	x
	psetex	✓	✓	✓	-
	setex	✓	✓	✓	-
	set	✓	✓	✓	-
	setbit	✓	✓	✓	-
	setnx	✓	✓	✓	-
	setrange	✓	✓	✓	-
	strlen	✓	✓	✓	-
	bitfield	x	✓	✓	-
Hyperloglog Family	pfadd	✓	✓	✓	-
	pfcount	✓	✓	✓	x
	pfmerge	✓	✓	✓	x
server Family	client list	✓	✓	✓	-
	client getname	✓	✓	✓	-
	client setname	✓	✓	✓	-
	dbsize	✓	✓	✓	-
	info	✓	✓	Custom	-
	time	✓	✓	✓	-
	lastsave	✓	✓	✓	-
	slowlog	✓	✓	Custom	-
	cluster keyslot	x	x	✓	-
	cluster nodes	x	x	✓	-

	cluster slots	x	x	✓	—
	cluster info	x	x	✓	—
	lolwut	x	x	x	—

Redis/Valkey Edition Partially Supported

Command Usage Examples

Last updated: 2026-03-17 20:49:53

Tencent Cloud Distributed Cache Cluster Architecture is compatible with smart clients such as JedisCluster. For compatibility with JedisCluster, Tencent Cloud Distributed Cache modifies the IP list returned by the supported commands, and the IP address of each node in the returned information is the instance's private IPv4 address.

CLUSTER NODES

CLUSTER NODES is used to get the information of each node in a Tencent Cloud Distributed Cache cluster, where each output line represents a node. Node information includes node ID, private IPv4 address and port, node role (master or replica), attributes, and assigned slots. For more details, see the [CLUSTER NODES command](#).

```
[ crs- | DB0 ] # cluster nodes
10. .45:6379@12666 myself,master - 0 1656297709000 6 connected 0-16383
f2f3
7cbd 10. .45:6379@12460 slave f2f3c387b9f. 0 1656297710000 6 connected
```

CLUSTER SLOTS

CLUSTER SLOTS is used to get the mapping relationship between cluster slots and Redis instances. Each returned result contains:

- Start slot range.
- End slot range.
- Information of the cluster's master node corresponding to the slot range, including the private IPv4 address, port, and node ID.
- Information of the first replica of the cluster's master node corresponding to the slot range.
- Information of the second replica.
- And so on in a similar manner until the information of all replicas is returned.

```
[ crs- | DB0 ] # cluster slots
1) 1) "0"
   2) "16383"
   3) 10. .45,6379,f2f3
   4) 10. .45,6379,7cbd
```

Use Cases of Custom Commands

Last updated: 2024-11-05 10:22:22

Through VIP encapsulation, Memory Edition (Cluster Architecture) delivers a user experience in cluster mode comparable to the standalone edition, making it much easier for use in different scenarios. However, in Ops scenarios, each node in the cluster needs to be accessed frequently to locate exceptions. In this case, the custom command feature can add a **node ID** parameter to the parameter list of the original command in the format of `COMMAND arg1 arg2 ... [node ID]` in order to easily get the information of the specified node. The node ID can be obtained from the **Node Management** page in the [TencentDB for Redis® console](#) or through the `cluster nodes` command.

Version Description

On proxy agent versions prior to v5.5.0, the node ID is required for the execution of custom commands, but it is unnecessary on v5.5.0 and later.

INFO

This command returns the information and statistics of a server.

Custom command format

```
info [section] [node ID]
```

Here, optional parameters can be used to select a specific part of the information:

- `server` : The general information of a Redis server.
- `clients` : The information of connected clients.
- `memory` : The information of memory usage.
- `persistence` : The information of RDB and AOF.
- `stats` : The general statistics.
- `replication` : The information of master/replica replication.
- `cpu` : The information of CPU usage.
- `commandstats` : The statistics of Redis commands.
- `cluster` : The information of a Redis cluster.
- `keyspace` : statistics of database

Optional parameters can also take the following values:

- `all` : Returns all the information.
- `default` : Returns the default information.

For more information, see [INFO Command](#).

Sample code

The following example runs the `INFO` command with `section` being `server`:

```
[ crs-r | DB0 ] # info server f2f3c3f
# Server
redis_version:4.3.0
redis_git_sha1:5f5e6086
redis_git_dirty:1
redis_build_id:52eb703ea1aa8bfd
redis_mode:cluster
os:Linux 3.10.107-1-tlinux2-0056 x86_64
arch_bits:64
multiplexing_api:epoll
atomicvar_api:atomic-builtin
gcc_version:4.8.5
process_id:22781
run_id:f42b93c
tcp_port:2666
uptime_in_seconds:7171266
uptime_in_days:83
hz:10
lru_clock:11714491
executable:/data/redis/app/redis-server-ignore-40026013-2666-1-ignore/./redis-server-ignore-40026013-2666-1-ignore
config_file:/data/redis/app/redis-server-ignore-40026013-2666-1-ignore/redis-server-ignore-40026013-2666-1-ignore_redis.conf
```

SLOWLOG

This command reads slow logs. It uses `SLOWLOG GET` to return the entries in slow logs. You can specify to return only the last N entries and pass other parameters to this command, such as `SLOWLOG GET 10`.

Custom command format

```
slowlog get [Redis node ID]
slowlog get [slow log quantity][Redis node ID]
```

Sample code

```
1 > slowlog get 49a
1) 1) (integer) 1
   2) (integer) 16
   3) (integer) 16978
   4) 1) "evalsha"
      2) "f6f2"
      3) "1"
      4) ""
      5) "6f3t"
      6) "proxy_commands"
      7) "0.8"
      8) "0"
      9) "1642647550"
      10) ""
      11) "1800"
   5) "9.248.236.209:25626"
   6) ""
2) 1) (integer) 0
   2) (integer) 1642647553
   3) (integer) 16954
   4) 1) "EXPIRE"
      2) "ProxyNodeIds::insid:{8},timestamp:1642647550"
      3) "1800"
   5) "?:0"
   6) ""
```

FLUSHDB

This command deletes all keys of the currently selected database. It will never fail.

Custom command format

```
flushdb [Redis node ID]
```

Sample code

```
cd-crs-rhxxxay.sql.tencentcdb.com:24894> flushdb
f2f3c387b9fab0e67af02039845c60278b13bed0
OK
```

PING

This command is often used to test whether the connection still exists or to measure the latency.

Custom command format

```
ping [message] [node ID]
```

Sample code

```
[ crs-rh**** | DB0 ] # PING "PONG" f2f3c3*****  
PONG  
[ crs-rh**** | DB0 ] # PING "hello world"  
hello world
```

KEYS

This command queries all the matched keys.

Custom command format

```
keys [pattern] [Redis node ID]
```

Sample code

```
keys a* f2f3c3*****
```

```
cd-crs-rl sql. com:24894> keys a*c f2f  
1) "avc"  
2) "azc"  
3) "abc"  
4) "acc"
```

SCAN

Custom command format

```
scan cursor [MATCH pattern] [COUNT count] [Redis node ID]
```

Sample code

```
[ crs-***** | DB0 ] # scan 0 f2f3c3*****  
1) "2"
```

IMonitor

Note:

- The iMonitor command requires the Redis proxy version of 5.6.0 or later.

- Native Redis does not support the iMonitor command, and redis-cli cannot recognize this command. To execute this command, you should use the telnet tool.

The command needs to be executed on the proxy node, and the parameter is the ID of the Redis shard node.

```
imonitor [Redis node ID]
```

Use Cases

```
imonitor 3dba154c67925520ef1a1e2c41d8cc22d7f4****
+OK
+1680504260.729707 [0 127.0.0.1:6379] "auth" ****
+1680504260.730070 [0 127.0.0.1:6379] "info" "commandstats"
+1680504262.243004 [0 127.0.0.1:6379] "AUTH" ****
```

Memcached Edition Command Compatibility

Last updated: 2026-03-17 18:11:00

Memcached Edition instances are compatible with the Memcached protocol and support the following commands: set, add, replace, append, prepend, cas, get, bget, gets, get_ext, gets_ext, delete, incr, decr.

Command	Compatible Version	Meaning
set	Memcached protocol 1.6	Set a key-value pair. If the key does not exist, it will create the key; if the key already exists, it will overwrite the old value.
add		Add a key-value pair to the cache, but only when the key does not exist. If the key already exists, however, the command does nothing.
replace		Replace a key-value pair in the cache, but only when the key already exists. If the key does not exist, however, the command does nothing.
append		Append data to the end of the existing value of a key. If the key does not exist, however, the command does nothing.
prepend		Prepend data to the beginning of the existing value of a key. If the key does not exist, however, the command does nothing.
cas		Compare and swap. This command first checks whether the current value of the key matches the provided value; if it does, the value is updated to the new one.
get		Retrieve the value of one or more keys from the cache. If no keys are found, an error will be returned.
bget		Batch Get. This is a binary protocol command used to retrieve the values of multiple keys from the cache.
gets		Similar to get, it obtains the value of a key in the cache and returns a version number.
get_ext		Extended binary protocol command that allows the client to specify more options to control the data retrieval process.
gets_ext	Extended command with versioning that allows the client to retrieve the value of a key while also obtaining a version number for subsequent atomic operations.	

delete	Delete a key–value pair from the cache. If the key does not exist, however, the command does nothing.
incr	Increment the value of a key by a specified integer. If the key does not exist, an initial value can be specified. If the value of the key is not an integer, the operation will fail.
decr	Decrement the value of the key by the specified integer.