|  |  |
| --- | --- |
| **Feature** | **Description** |
| Master-Slave Replication | It allows replica Redis instances to be exact copies of master instances. The replica will automatically reconnect to the master every time the link breaks, and will attempt to be an exact copy of it regardless of what happens to the master. |
| Cluster | Redis Cluster provides a way to run a Redis installation where data is automatically sharded across multiple Redis nodes. Redis Cluster also provides some degree of availability during partitions—in practical terms, the ability to continue operations when some nodes fail or are unable to communicate. |
| Sentinel | Redis Sentinel provides high availability for Redis when not using [Redis Cluster](https://redis.io/docs/latest/operate/oss_and_stack/management/scaling/).Redis Sentinel also provides other collateral tasks such as monitoring, notifications and acts as a configuration provider for clients. |
| Persistence  (RDB、AOF) | Persistence refers to the writing of data to durable storage, such as a solid-state disk (SSD). Redis provides a range of persistence options. These include:  RDB(Redis Database): RDB persistence performs point-in-time snapshots of your dataset at specified intervals.  AOF(Append Only File): AOF persistence logs every write operation received by the server. These operations can then be replayed again at server startup, reconstructing the original dataset. Commands are logged using the same format as the Redis protocol itself. |
| Command/  Redis-cli | The command implements the interaction between the user and the server and some other operations and it is rich, covering many aspects such as data operation, data persistence, and server management,Redis-cli is the Redis command line interface. |
| Lua Scripting | Redis lets users upload and execute Lua scripts on the server. Scripts can employ programmatic control structures and use most of the [commands](https://redis.io/docs/latest/commands/) while executing to access the database. Because scripts execute in the server, reading and writing data from scripts is very efficient. |
| Datastucture | At its core, Redis provides a collection of native data types that help you solve a wide variety of problems.[Redis Community Edition](https://redis.io/docs/latest/operate/oss_and_stack/) implements the following data types:[String](https://redis.io/docs/latest/develop/data-types/" \l "strings)/[Hash](https://redis.io/docs/latest/develop/data-types/" \l "hashes)/[List](https://redis.io/docs/latest/develop/data-types/" \l "lists)/[Set](https://redis.io/docs/latest/develop/data-types/" \l "sets)/[Sorted set](https://redis.io/docs/latest/develop/data-types/" \l "sorted-sets)/[Stream](https://redis.io/docs/latest/develop/data-types/" \l "streams)/[Bitmap](https://redis.io/docs/latest/develop/data-types/" \l "bitmaps)/[Bitfield](https://redis.io/docs/latest/develop/data-types/" \l "bitfields)/[Geospatial](https://redis.io/docs/latest/develop/data-types/" \l "geospatial-indexes) |
| Networking | Based on the event-driven single thread model, I/O multiplexing technology is used to realize efficient network communication, which provides a strong guarantee for the high performance of Redis. |
| Module | Redis modules make it possible to extend Redis functionality using external modules, rapidly implementing new Redis commands with features similar to what can be done inside the core itself. |
| Redis Caching | Client-side caching reduces network traffic between a Redis client and the server, which generally improves performance. |
| Publish/  Subscribe | Redis has a [publish/subscribe](https://redis.io/docs/latest/develop/interact/pubsub/) (Pub/sub) feature that implements the well-known [design pattern](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) of the same name. You can publish messages from a particular client connection to a channel maintained by the server. Other connections that have subscribed to the channel will receive the messages in the order you sent them. Use pub/sub to share small amounts of data among clients easily and efficiently. |
| Access Control Lists | The Redis ACL, short for Access Control List, is the feature that allows certain connections to be limited in terms of the commands that can be executed and the keys that can be accessed. |
| Cross-platform Portability | This allows Redis to run on different operating systems and hardware architectures without extensive modification or recompilation. |
| Active Defragmentation | Automatic memory defragmentation introduced in Redis 4.0 and above improves memory utilization. |