Concepts and architecture

A Redis Enterprise cluster is composed of identical nodes that are deployed within a data center or stretched across local availability zones. Redis Enterprise architecture is made up of a management path (shown in the blue layer in Figure 1 below) and data access path (shown in the red layer in Figure 1 below).

- Management path includes the cluster manager, proxy and secure REST API/UI for programmatic administration. In short, cluster manager is responsible for orchestrating the cluster, placement of database shards as well as detecting and mitigating failures. Proxy helps scale connection management.

- Data Access path is composed of master and replica Redis shards. Clients perform data operations on the master shard. Master shards maintain replica shards using the in-memory replication for protection against failures that may render master shard inaccessible.

![Figure 1 Redis Enterprise Nodes with blue layer representing the management path and red tiles representing the data access path with Redis as the shards.](image)

High availability with Redis Enterprise

Redis Enterprise uses in-memory replication to maintain master and replicas. Redis Enterprise comes with various watchdogs that detect and protect against many failures types. Under failures such as node, network, process failures that render master replica inaccessible, Redis Enterprise automatically promotes the replica to be a master replica and redirects the client connection transparently to the new master replica.

Besides the intra-cluster replication, Redis Enterprise also has built-in WAN-based replication for Redis deployments across multiple data centers. WAN-based replication mechanisms in Redis Enterprise are designed, protect against total Data Center or wider network failures.

 Scaling databases

Each Redis Enterprise cluster can contain multiple databases. In Redis, databases represent data that belong to a single application, tenant, or microservice. Redis Enterprise is built to scale to 100s of databases per cluster to provide flexible and efficient multi-tenancy models.

Each database can contain few or many Redis shards. Sharding is transparent to Redis applications. Master shards in the database process data operations for a given subset of keys. The number of shards per database is configurable and depend on the throughput needs of the applications. Databases in Redis Enterprise can be resharded into more Redis shards to scale throughput while maintaining sub-millisecond latencies. Resharding is performed without downtime.
In Redis Enterprise, each database has a quota of RAM. The quota cannot exceed the limits of the RAM available on the node. However, with Redis Enterprise Flash, RAM is extended to the local flash drive (SATA, NVMe SSDs etc). The total quota of the database can take advantage of both RAM and Flash drive. The administrator can choose the RAM vs Flash ratio and adjust that anytime in the lifetime of the database without downtime.

With Redis on Flash, instead of storing all keys and data for a given shard in RAM, less frequently accessed values are pushed to flash. If applications need to access a value that is in flash, Redis Enterprise automatically brings the value into RAM. Depending on the flash hardware in use, applications experience slightly higher latency when bringing values back into RAM from flash. However subsequent accesses to the same value is fast, once the value is in RAM.

Data durability with Redis Enterprise

Redis Enterprise has two durability options:

- Disk-based durability: Redis Enterprise still maintains a durable copy on disk. Just like disk-based systems, this IO path is placed on a slower and durable network-attached storage device. Redis databases provide tunable options to maintain this durable copy and keep it up to date with frequent periodic writes all the way to every write operation.

- Replication-based durability: Redis Enterprise also maintains a replica, a slave shard, for durability. This replicated durability protects against node, rack, or zone failures. Replicated-durability provides better write performance over network-attached storage writes. This means under an unplanned interruption, it is more likely that your replica is more up to date as compared to your durable copy on disk. To take full advantage of the replicated-durability, Redis provides the `WAIT` command. `WAIT` makes sure that a write can wait for acknowledgment until multiple replicas confirm that write. This makes sure that a write confirmed with `WAIT` on replicas are durable even if a node catches on fire and never comes back to the cluster.

Updated: September 23, 2021