# cluster-failover


- FORCE option: manual failover when the primary is down
- TAKEOVER option: manual failover without cluster consensus
- Implementation details and notes


This command, that can only be sent to a Valkey Cluster replica node, forces the replica to start a manual failover of its primary instance.

A manual failover is a special kind of failover that is usually executed when there are no actual failures, but we wish to swap the current primary with one of its replicas (which is the node we send the command to), in a safe way, without any window for data loss. It works in the following way:

1. The replica tells the primary to stop processing queries from clients.
2. The primary replies to the replica with the current *replication offset*.
3. The replica waits for the replication offset to match on its side, to make sure it processed all the data from the primary before it continues.
4. The replica starts a failover, obtains a new configuration epoch from the majority of the primaries, and broadcasts the new configuration.
5. The old primary receives the configuration update: unblocks its clients and starts replying with redirection messages so that they'll continue the chat with the new primary.

This way clients are moved away from the old primary to the new primary atomically and only when the replica that is turning into the new primary has processed all of the replication stream from the old primary.

There is a built-in timeout for the primary to resume processing queries if the replica does not catch quickly enough or is unable to secure the votes to take over. The default failover timeout is 5000ms. It is possible to configure the timeout via the `cluster-manual-failover-timeout` configuration parameter (added in Valkey 8.1) and decide how long the primary will pause in the worst case scenario.

# FORCE option: manual failover when the primary is down

The command behavior can be modified by two options: **FORCE** and **TAKEOVER**.

If the **FORCE** option is given, the replica does not perform any handshake with the primary, that may be not reachable, but instead just starts a failover ASAP starting from point 4. This is useful when we want to start a manual failover while the primary is no longer reachable.

However using **FORCE** we still need the majority of primaries to be available in order to authorize the failover and generate a new configuration epoch for the replica that is going to become primary.

# TAKEOVER option: manual failover without cluster consensus

There are situations where this is not enough, and we want a replica to failover without any agreement with the rest of the cluster. A real world use case for this is to mass promote replicas in a different data center to primaries in order to perform a data center switch, while all the primaries are down or partitioned away.

The **TAKEOVER** option implies everything **FORCE** implies, but also does not uses any cluster authorization in order to failover. A replica receiving `CLUSTER FAILOVER TAKEOVER` will instead:

1. Generate a new `configEpoch` unilaterally, just taking the current greatest epoch available and incrementing it if its local configuration epoch is not already the greatest.
2. Assign itself all the hash slots of its primary, and propagate the new configuration to every node which is reachable ASAP, and eventually to every other node.

Note that **TAKEOVER violates the last-failover-wins principle** of Valkey Cluster, since the configuration epoch generated by the replica violates the normal generation of configuration epochs in several ways:

1. There is no guarantee that it is actually the higher configuration epoch, since, for example, we can use the **TAKEOVER** option within a minority, nor any message exchange is performed to generate the new configuration epoch.
2. If we generate a configuration epoch which happens to collide with another instance, eventually our configuration epoch, or the one of another instance with our same epoch, will be moved away using the *configuration epoch collision resolution algorithm*.

Because of this the **TAKEOVER** option should be used with care.

# Implementation details and notes

- `CLUSTER FAILOVER`, unless the **TAKEOVER** option is specified, does not execute a failover synchronously. It only *schedules* a manual failover, bypassing the failure detection stage.
- An `OK` reply is no guarantee that the failover will succeed.
- A replica can only be promoted to a primary if it is known as a replica by a majority of the primaries in the cluster. If the replica is a new node that has just been added to the cluster (for example after upgrading it), it may not yet be known to all the primaries in the cluster. To check that the primaries are aware of a new replica, you can send `CLUSTER NODES` or `CLUSTER REPLICAS` to each of the primary nodes and check that it appears as a replica, before sending `CLUSTER FAILOVER` to the replica.
- To check that the failover has actually happened you can use `ROLE`, `INFO REPLICATION` (which indicates "role:master" after successful failover), or `CLUSTER NODES` to verify that the state of the cluster has changed sometime after the command was sent.
- To check if the failover has failed, check the replica's log for "Manual failover timed out", which is logged if the replica has given up after a few seconds.

---

□□□ #1
□ □□□ □□□ 2026-10-31 15:11:45 UTC
□ □□□ □□□ 2026-10-31 15:11:45 UTC