failover

- Optional arguments
- Failover abort

This command will start a coordinated failover between the currently-connected-to primary and one of its replicas. The failover is not synchronous, instead a background task will handle coordinating the failover. It is designed to limit data loss and unavailability of the cluster during the failover. This command is analogous to the `CLUSTER FAILOVER` command for non-clustered Valkey and is similar to the failover support provided by sentinel.

The specific details of the default failover flow are as follows:

1. The primary will internally start a `CLIENT PAUSE WRITE`, which will pause incoming writes and prevent the accumulation of new data in the replication stream.
2. The primary will monitor its replicas, waiting for a replica to indicate that it has fully consumed the replication stream. If the primary has multiple replicas, it will only wait for the first replica to catch up.
3. The primary will then demote itself to a replica. This is done to prevent any dual primary scenarios. NOTE: The primary will not discard its data, so it will be able to rollback if the replica rejects the failover request in the next step.
4. The previous primary will send a special PSYNC request to the target replica, `PSYNC FAILOVER`, instructing the target replica to become a primary.
5. Once the previous primary receives acknowledgement the `PSYNC FAILOVER` was accepted it will unpause its clients. If the PSYNC request is rejected, the primary will abort the failover and return to normal.

The field `master_failover_state` in `INFO replication` can be used to track the current state of the failover, which has the following values:

- `no-failover`: There is no ongoing coordinated failover.
- `waiting-for-sync`: The primary is waiting for the replica to catch up to its replication offset.
- `failover-in-progress`: The primary has demoted itself, and is attempting to hand off ownership to a target replica.

NOTE: During the `failover-in-progress` phase, the primary first demotes itself to a replica and then notifies the replica to promote itself to primary. These two steps are an asynchronous process, which may result in the simultaneous existence of two nodes as replicas. In this scenario, for clients

that support REDIRECT (explicitly execute `CLIENT CAPA REDIRECT`), the redirection result may bounce back and forth between the two replicas until the target replica completes the process of promoting itself to primary. To avoid this situation, during the `failover-in-progress` phase, we temporarily suspend the clients that need to be redirected until the replica truly becomes the primary, and then resume the execution.

If the previous primary had additional replicas attached to it, they will continue replicating from it as chained replicas. You will need to manually execute a `REPLICAOF` on these replicas to start replicating directly from the new primary.

Optional arguments

The following optional arguments exist to modify the behavior of the failover flow:

- `TIMEOUT` *milliseconds* -- This option allows specifying a maximum time a primary will wait in the `waiting-for-sync` state before aborting the failover attempt and rolling back. This is intended to set an upper bound on the write outage the Valkey cluster can experience. Failovers typically happen in less than a second, but could take longer if there is a large amount of write traffic or the replica is already behind in consuming the replication stream. If this value is not specified, the timeout can be considered to be "infinite".
- `TO` *HOST PORT* -- This option allows designating a specific replica, by its host and port, to failover to. The primary will wait specifically for this replica to catch up to its replication offset, and then failover to it.
- `FORCE` -- If both the `TIMEOUT` and `TO` options are set, the force flag can also be used to designate that that once the timeout has elapsed, the primary should failover to the target replica instead of rolling back. This can be used for a best-effort attempt at a failover without data loss, but limiting write outage.

NOTE: The primary will always rollback if the `PSYNC FAILOVER` request is rejected by the target replica.

Failover abort

The failover command is intended to be safe from data loss and corruption, but can encounter some scenarios it can not automatically remediate from and may get stuck. For this purpose, the `FAILOVER ABORT` command exists, which will abort an ongoing failover and return the primary to its normal state. The command has no side effects if issued in the `waiting-for-sync` state but can introduce multi-primary scenarios in the `failover-in-progress` state. If a multi-primary scenario is encountered, you will need to manually identify which primary has the latest data and designate it as the primary and have the other replicas.

NOTE: `REPLICAOF` is disabled while a failover is in progress, this is to prevent unintended interactions with the failover that might cause data loss.

□□□ #1

□ □□□ □□□ 2026-10-31 15:13:16 UTC

□ □□□ □□□ 2026-10-31 15:13:16 UTC