

# KeyDB (Redis compatible) with Multi-master or Replica mode

If you can't afford for your database to be down for even a few minutes, you need a Multi-Master cluster to ensure high availability. This means that one node can be taken offline (e.g. for maintenance or upgrade purposes) without impacting availability, as the other node will continue to serve production traffic. Further, it doubles your capacity to read or write to the database and provides an additional layer of protection against data loss.

KeyDB is a fork of Redis that brings multithreading and Multi-Master replication, so you can have a highly available cluster of Redis in-memory DB. Usually setting up a cluster is a non-trivial task but in OpenVM you can do this in a few clicks.

To begin, you will need to have deployed two KeyDB instances













1) Go to elestio Dashboard > Deploy new service> Databases > select KeyDB, scroll down and name it for example **keydb-1** then click on the "Create service button"

Create Service

1 Select service 2 Select provider, region & service plan 3 Select Support & advanced setting

Databases Applications Development Hosting & Infra Full Stack CI/CD All

Search service by name

 <b>PostgreSQL</b> PostgreSQL is a powerful, open-source object-relational database system, known for reliability, data integrity and performance.	 <b>MySQL</b> MySQL is an Oracle-backed open-source RDBMS that runs on almost all platforms.	 <b>MariaDB</b> The open source relational database
 <b>MongoDB</b> MongoDB is a document-oriented NoSQL database used for high-volume data storage.	 <b>Redis</b> Redis is an open-source, in-memory database, cache and message broker.	 <b>KeyDB</b> KeyDB is both your cache and database, for cloud-optimized solutions. <a href="#">Details</a> <a href="#">Select</a>
 <b>TimescaleDB</b> TimescaleDB is the leading open-source relational database with support for time-series data.	 <b>ClickHouse</b> ClickHouse is an open-source, column-oriented DBMS for online analytical processing.	 <b>ScyllaDB</b> ScyllaDB is a true NoSQL database for the most demanding applications.
 <b>InfluxDB</b> InfluxDB is a scalable datastore that empowers developers to build IoT, analytics and monitoring software.	 <b>MSSQL</b> SQL Server 2019 is a modern data platform designed to tackle the challenges of today's data professional.	 <b>Neo4j</b> Neo4j is the world's leading Graph Database

2) Again, go to elestio Dashboard > Deploy new service> Databases > select KeyDB, scroll down and name it for example **keydb-2** then click on the "Create service button"

3) Wait for the 2 instances to be ready

4) In the elestio dashboard open the service details of **keydb-1** and click on the "Configure cluster" button

Server type: SMALL-1C-2G (1 VCPU - 2 GB RAM - 20 GB storage) Provider: hetzner

### Configure Cluster ✕

Select a partner instance from the list

▼

Or indicate details from an external instance

**Cluster Mode** ☐ Disabled ☐ Replica ☐ Multi Master

If you selected "Multi-Master" replication, you will need to do the same cluster configuration from the other node with this node infos.

5) Select in the partner instance dropdown "**keydb-2**" as the partner, then select "Multi-Master" in Cluster mode, then click on "Apply changes" button

6) In the elestio dashboard open the service details of **keydb-2** and click on the "Configure cluster" button

7) Select in the partner instance dropdown "**keydb-1**" as the partner, then select "Multi-Master" in Cluster mode, then click on "Apply changes" button

**All done. You now have a multi-master KeyDB cluster.**

You can now read and write on both instances. If instance A is down you will still be able to use instance B and vice versa. Also, if you restore a backup on one instance it will be automatically replicated to the other instance.

## How to use Multi-Master cluster from Node.js

```
////////// NodeJS sample //////////
const Redis = require("ioredis");
const cluster = new Redis.Cluster([
  { port: 23647, password:'FIRST_INSTANCE_PASSWORD_HERE', host: "Type_your_first_node_ip_here"
  },
  { port: 23647, password:'FIRST_INSTANCE_PASSWORD_HERE', host:
  "Type_your_second_node_ip_here" }
]);

cluster.set("foo", "bar");
cluster.get("foo", (err, res) => {
  // res === 'bar'
});
////////// //////////
```

## How to test your High Availability Cluster

1. Shut down one of the VMs (instance A). You should still be able to connect, read and write on your cluster.
2. Restart instance A, wait 30 seconds, then shut down instance B.
3. Test your connectivity and read/write access to the cluster again.
4. Finally, restart instance B.

## Use Redis Insight to test your cluster

1. Open the service details and click on Admin UI to get url and credentials of Redis Insight.
2. Open a browser tab with the Admin UI for instance B.
3. Open another browser tab for instance A.

4. Go to Local Redis > Browser > Add Key.
5. Create a key, of type String, named 'A', with value 100, then click on Add.
6. Go to the other browser tab for instance A and select Local Redis > Browser > and check if you see key A with the correct value.
7. You can also test it by modifying the A key - e.g. set another value, or by creating a new key and checking in your first tab if the change has been correctly replicated.

---

Revision #5

Created 19 February 2022 17:18:27 by Joseph Benguira

Updated 22 July 2024 14:07:26 by Amit