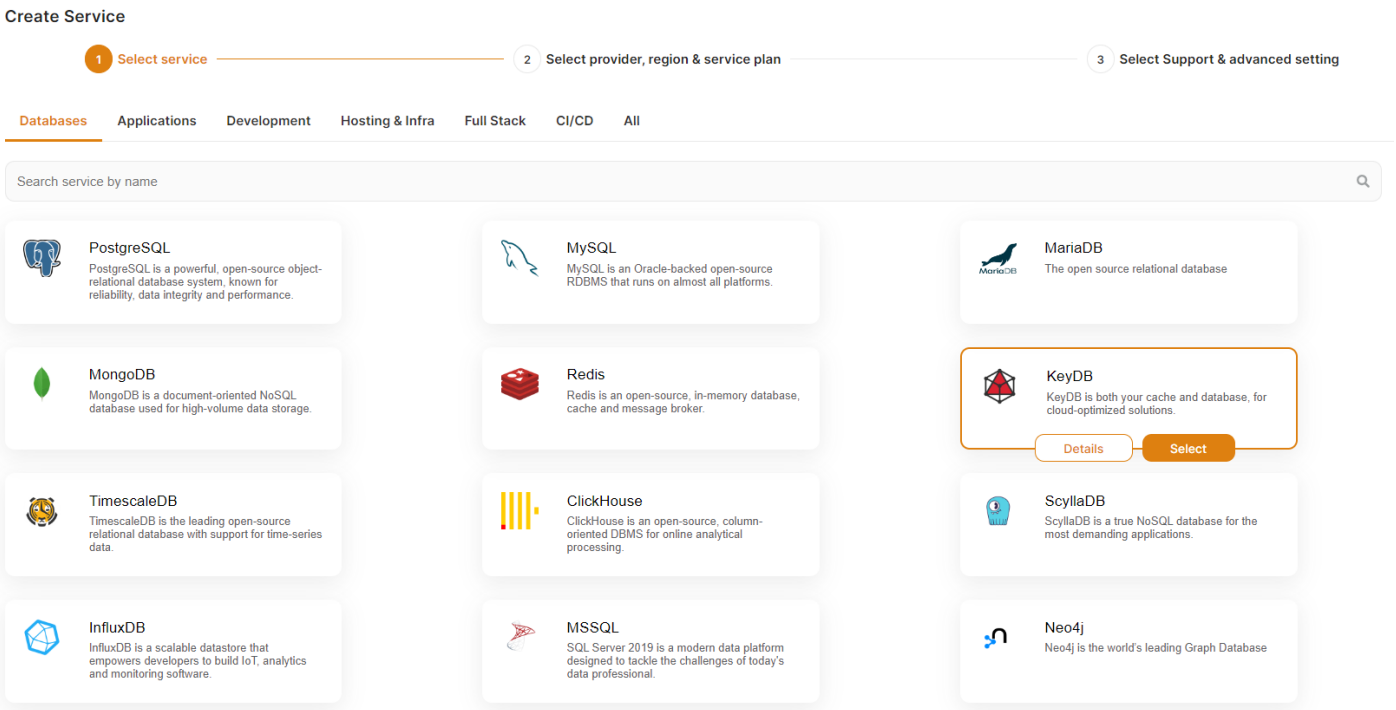
KeyDB (Redis compatible) with Multi-master or Replica mode

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

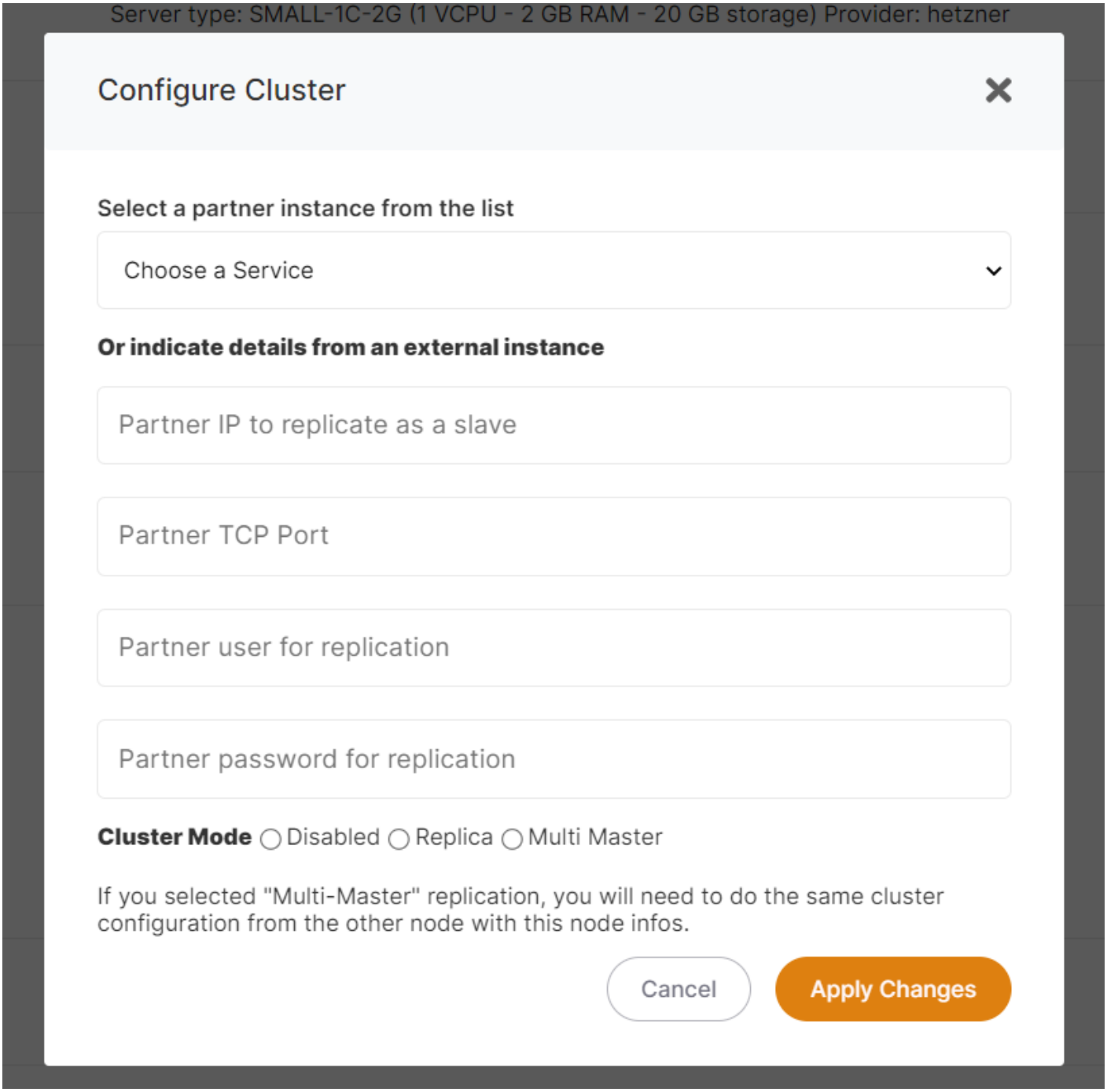
1) Go to elestio Dashboard > Deploy new service> Databases > select KeyDB, scroll down and name it for example **keydb-1** then click on the "Create service button"



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



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 29 September 2022 12:36:24 by Amit Shukla