Getting Started with Redis Enterprise Active-Active Databases

In this guide, we'll set up an Active-Active database, formerly known as CRDB (Conflict-free Replicated DataBase) spanning across two Redis Enterprise Software clusters for test and development environments. Here are the steps:

- Step 1: Run two Redis Enterprise Software (RS) Docker containers
- Step 2: Set up each container as a cluster
- Step 3: Create a new Redis Enterprise Active-Active database
- Step 4: Test connectivity to the Active-Active database

To run an Active-Active database on installations from the RS download package, set up two RS installations and continue from Step 2.

Note: This getting started guide is for development or demonstration environments. To set up an Active-Active database in a production environment, use the instructions for creating an Active-Active database.

### Step 1: Run two containers

To spin up two RS containers, run these commands:

```
docker run -d --cap-add sys_resource -h rp1_node1 --name rp1_node1 -p 8443:8443 -p 9443:9443 -p 12000:12000 redislabs/redis

docker run -d --cap-add sys_resource -h rp2_node1 --name rp2_node1 -p 8445:8443 -p 9445:9443 -p 12002:12000 redislabs/redis
```

The `-p` options map the admin console port (8443), REST API port (9443), and database access port differently for each container to make sure that all containers can be accessed from the host OS that is running the containers.

### Step 2: Setup two clusters

1. For cluster 1, direct your browser to `https://localhost:8443` on the host machine to see the Redis Enterprise Software web console. Simply click the Setup button on the page to get started.

Note: Depending on your browser, you may see a certificate error. Continue to the website.
2. On the **node configuration** page, select your default settings and provide a cluster FQDN, for example `cluster1.local`. Then click **Next** button.

3. If you don’t have a license key, click the **Next** button to try the trial version of the product.

4. On the next screen, set up a Cluster Administrator account using an email for the login and a password.
Repeat the same operations for cluster 2 with these differences:

- In your web browser, go to https://localhost:8445 to set up the cluster 2.
- For the Cluster name (FQDN), enter a different name, such as cluster2.local.

Now we have two Redis Enterprise Software clusters with FQDNs cluster1.local and cluster2.local.

Note: Each Active-Active instance must have a unique fully-qualified domain name (FQDN).

Step 3: Create a Redis Active-Active database

1. After you login to cluster1.local, select the Redis database and deployment type Geo-Distributed. Then click Next.
2. In create database, click the show advanced option and:

1. For the database name, enter: database1
2. For the endpoint port number, enter: 12000
3. In the participating clusters list, add the address and admin credentials for:
   - https://cluster1.local:9443 - the cluster you are currently connected to
   - https://cluster2.local:9443 - the other cluster
4. In Database clustering, either:
   - Make sure that Database clustering is enabled and select the number of shards that you want to have in the database. When database clustering is enabled, databases are subject to limitations on Multi-key commands. You can increase the number of shards in the database at any time.
   - Clear Database clustering to use only one shard and to avoid Multi-key command limitations.

   **Note:** You cannot enable or disable database clustering after the Active-Active database is created.

3. Click Activate to create your Active-Active database.
After the Active-Active database is created, access the RS admin console of cluster 1 at https://localhost:8443 and of cluster 2 at https://localhost:8445.

Make sure that each cluster has an Active-Active database member database with the name `database1`.

In a real-world deployment, cluster 1 and cluster 2 would most likely be in separate data centers in different regions. However, for local testing we created the scale-minimized deployment using two local clusters running on the same host.

**Step 4: Test the connection to your member Redis Active-Active databases**

With the Redis database created, you are ready to connect to your database to store data. You can use one of the following ways to test connectivity to your database:

- Connect with `redis-cli`, the built-in command-line tool
- Connect with a *Hello World* application written in Python

Remember we have two member Active-Active databases that are available for connections and concurrent reads and writes. The member Active-Active databases are using bi-directional replication to for the global Active-Active database.
Connecting using redis-cli

redis-cli is a simple command-line tool to interact with redis database.

1. To switch your context into the RS container of node 1 in cluster 1, run:

   docker exec -it rp1_node1 bash

2. To use redis-cli on port 12000, run:

   redis-cli -p 12000

3. Store and retrieve a key in the database to test the connection with these commands:

   - set key1 123
   - get key1

   The output of the command looks like this:

   127.0.0.1:12000> set key1 123
   OK
   127.0.0.1:12000> get key1
   "123"

4. Enter exit to exit the redis-cli context and enter exit again to exit the RS container of node 1 in cluster 1.

5. To see that the key replicated to cluster 2, repeat the steps to switch your context into the RS container of node 1 in cluster 2, run the redis-cli and retrieve key1.

   The output of the commands looks like this:
$ docker exec -it rp2_node1 bash
$ redis-cli -p 12000
127.0.0.1:12000> get key1
"123"

**Connecting using Hello World application in Python**

A simple python application running on the host machine can also connect to the database.

**Note:** Before you continue, you must have python and redis-py (python library for connecting to Redis) configured on the host machine running the container.

1. In the command-line terminal, create a new file called `redis_test.py`.

```
vi redis_test.py
```

2. Paste this code into the `redis_test.py` file.

This application stores a value in key1 in cluster 1, gets that value from key1 in cluster 1, and gets the value from key1 in cluster 2.

```python
import redis

rp1 = redis.StrictRedis(host='localhost', port=12000, db=0)
rp2 = redis.StrictRedis(host='localhost', port=12002, db=0)

print("set key1 123 in cluster 1")
print(rp1.set('key1', '123'))
print("get key1 cluster 1")
print(rp1.get('key1'))

print("get key1 from cluster 2")
print(rp2.get('key1'))
```

3. To run the “redis_test.py” application, run:

```
python redis_test.py
```

If the connection is successful, the output of the application looks like:

```
set key1 123 in cluster 1
True
get key1 cluster 1
"123"
get key1 from cluster 2
"123"
```

**Updated:** December 6, 2021