Creating a Redis Enterprise Software Database

You can create Redis databases that are clustered and distributed across a single Redis Enterprise Software (RS) cluster. These databases can use Redis Enterprise features like:

- Redis on Flash
- High availability
- Data persistence
- Redis modules

You can create databases according to the number of shards in your subscription and the memory available on the machine.

**Note:** For databases with Active-Active replication for geo-distributed locations, create an Active-Active database.

Creating a new redis database

To create a new database:

1. In your web browser, open the admin console of the cluster that you want to connect to in order to create the database.

   By default, the address is: `https://<RS_address>:8443`

2. In databases, click.

   If you do not have any databases on the node, you are prompted to create a database.

3. Click Next to create a single-region, in-memory database.

   If your cluster supports Redis on Flash (RoF), in Runs on you can select Flash so that your database uses Flash memory.
4. Enter the mandatory details of the new database:

- **Name** - The database name requirements are:
  - Maximum of 63 characters
  - Only letter, number or hyphen (-) characters
  - Starts with a letter; ends with a letter or digit.

  **Note:** The database name is not case-sensitive

- **Memory limit** - The database memory limits includes all database replicas and shards, including slave shards in database replication and database shards in database clustering. If the total size of the database in the cluster reaches the memory limit, then the data eviction policy for the database is enforced.

  **Note:** If you create a Redis on Flash or a Memcached Flash database, you also have to set the RAM-to-Flash ratio for this database. Minimum RAM portion is 10%, and maximum RAM portion is 50%.

5. Configure the database options that you want for the database:

- **Replication** - We recommend that you use intra-cluster replication to create slave shards for each database for high-availability of your data.

  If the cluster is configured to support rack-zone awareness, you can also enable rack-zone awareness for the database.

- **Redis Modules** - When you create a new in-memory database, you can enable multiple Redis modules to the database. For RoF databases, you can add modules that support RoF.

  **Note:** Modules can be added to the database only when creating a new database. You can’t add a module to an existing database.
To add a module to the database:

1. In the Redis Modules field, click 🔄.
2. Select the module that you want to add.
3. If you want the module to use a custom configuration, click Add configuration and enter the optional custom configuration.
4. Click 🔄.

- **Data persistence** - To protect against loss of data stored in RAM, you can enable data persistence and select to store a copy of the data on disk with snapshots or Append Only File.

- **Default database access** - When you configure a password for your database, all connections to the database must authenticate with the AUTH command. If you also configure an access control list, connections can specify other users for authentication, and requests are allowed according to the Redis ACLs specified for that user.

Note that creating a database without further ACLs (see below) contains a default user with full access to the database which in turn requires the definition a password for security reasons.

6. Configure the database advanced options that you want for the database:

- **Access Control List** - You can specify the user roles that have access to the database and the Redis ACLs that apply to those connections.

To define an access control list:

1. In the Access control list section of the database configuration, click 🔄.
2. Select the role that you want to have access to the database.
3. Select the ACL that you want the role to have in the database.
4. Click Save to save the ACL.
5. Click Update to save the changes to the database.

- **Endpoint port number** - You can define the port number that clients use to connect to the database, or a port is randomly selected.

  Note: You cannot change the port number after the database is created.

- **Database clustering** - You can either:

  - Enable database clustering 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.

    You can accept the standard hashing policy or define a custom hashing policy to define where keys are located in the clustered database.

  - Clear the Database clustering option to use only one shard so that you can use Multi-key commands without the limitations.

- **OSS Cluster API** -

Redis OSS Cluster API reduces access times and latency with near-linear scalability. The Redis OSS Cluster API provides a simple mechanism for Redis clients to know the cluster topology.

Clients must first connect to the master node to get the cluster topology, and then they connect directly to the Redis
proxy on each node that hosts a master shard.

**Note:** You must use a client that supports the OSS cluster API to connect to a database that has the OSS cluster API enabled.

- **Data eviction policy** - By default, when the total size of the database reaches its memory limit the database evicts keys according to the least recently used keys out of all keys with an "expire" field set in order to make room for new keys. You can select a different data eviction policy.

- **Replica Of** - You can make this database a repository for keys from other databases.

- **TLS** - You can require TLS encryption and authentication for all communications, TLS encryption and authentication for Replica Of communication only, and TLS authentication for clients.

- **Periodic backup** - You can configure periodic backups of the database, including the interval and backup location parameters.

- **Alerts** - You can select alerts to show in the database status and configure their thresholds. You can also select to send the alerts by email to relevant users.

1. Click **Activate**.

2. If you did not specify a port number for the database, note the port number shown in the Endpoint field of the database configuration.

Simple connectivity test

Once the database is created, you can find the endpoint and port for the database in the admin console on the configuration page of each database. It is listed under the “Endpoint” property.

There are a few simple ways to check connectivity to your database:

- telnet
- Redis_cli
- Using a simple application
Testing database connectivity with Telnet

As a quick smoke test, telnet to your endpoint and port for the database. Then run the redis PING command.

Using the URL-based connection method, the result looks like:

```
$ telnet redis-19836.c9.us-east-1-2.ec2.cloud.redislabs.com 19836
Trying 54.89.217.184…
Escape character is '^]'.
AUTH my_redis_password
PING
+PONG
```

Testing database connectivity with redis cli

The redis-cli utility is installed on each node in RS cluster with the rest of the utilities.

Using the URL-based connection method, you can connect the database endpoint URL and port number. The result looks like:

```
$ redis-cli -h redis-19836.c9.us-east-1-2.ec2.cloud.redislabs.com -p 19836 -a astrongpassword
redis-19836.c9.us-east-1-2.ec2.cloud.redislabs.com:19836> PING
PONG
```

Testing database connectivity with a simple application

You can also use a simple application to test connectivity to your database. Here is a simple python app that connects to the database by IP address. The app uses the discovery service that is compliant with Redis Sentinel API.

In the IP-based connection method, you only need the database name, not the port number. Here we simply use the discovery service that listens on port 8001 on all nodes of the cluster to discover the endpoint for the database named "db1".

```
from redis.sentinel import Sentinel

# with IP based connections, a list of known node IP addresses is constructed
# to allow connection even if any one of the nodes in the list is unavailable.
sentinel_list = [
    ('10.0.0.44', 8001),
    ('10.0.0.45', 8001),
    ('10.0.0.45', 8001)
]

# change this to the db name you want to connect
db_name = 'db1'

sentinel = Sentinel(sentinel_list, socket_timeout=0.1)
r = sentinel.master_for(db_name, socket_timeout=0.1)

# set key "foo" to value "bar"
print r.set('foo', 'bar')
# set value for key "foo"
print r.get('foo')
```

In the URL-based connection method, you need to specify the endpoint and the port number for your database.
import redis

# the URL provided to redis. Redis method comes from the database configuration
# property called "Endpoint". The endpoint URL generated by the database is a
# combination of the cluster name (FQDN) and database port number.

r = redis.Redis(
    host='redis-19836.c9.us-east-1-2.ec2.cloud.redislabs.com',
    port=19836)

# set key "foo" to value "bar"
print(r.set('foo', 'bar'))

# set value for key "foo"
print(r.get('foo'))