



Redis Enterprise Software

Table of contents

- [Install and upgrade](#)
 - [Quickstarts](#)
 - [Quickstart](#)
 - [Docker quickstart](#)
 - [Install](#)
 - [Plan deployment](#)
 - [Hardware requirements](#)
 - [Supported platforms](#)
 - [Persistent node storage](#)
 - [File locations](#)
 - [AWS EC2 configuration](#)
 - [Prepare to install](#)
 - [Download installation package](#)
 - [Ensure port availability](#)
 - [Prepare Auto Tiering](#)
 - [Install on Linux](#)
 - [Installation script options](#)
 - [Manage install questions](#)
 - [Customize install locations](#)
 - [Customize user and group](#)
 - [Offline installation](#)
 - [Upgrade](#)
 - [Upgrade cluster](#)
 - [Upgrade database](#)
 - [Active-Active databases](#)
 - [Uninstall](#)
 - [Additional configuration](#)
 - [CentOS/RHEL firewall](#)
 - [Linux swap configuration](#)
 - [Socket file location](#)
 - [Product lifecycle](#)
 - [Create support package](#)
 - [Databases](#)
 - [Create a database](#)
 - [Configure](#)
 - [OSS Cluster API](#)
 - [Persistence](#)
 - [Proxy policy](#)
 - [Replica high availability](#)
 - [Shard placement](#)
 - [Connect](#)
 - [Supported connection clients](#)
 - [Test connection](#)
 - [Import and export](#)
 - [Import data](#)
 - [Export data](#)
 - [Flush database](#)
 - [Schedule backups](#)
 - [Migrate to Active-Active](#)
 - [Replica Of](#)
 - [Create Replica Of database](#)
 - [Troubleshoot repeat failures](#)
 - [Recover](#)
 - [Delete](#)
 - [Active-Active](#)
 - [Get started](#)
 - [Planning considerations](#)
 - [Create](#)
 - [Connect](#)
 - [Manage](#)
 - [Delete](#)
 - [Causal consistency](#)

- Distributed synchronization
- Syncer process
- Develop applications
 - Develop for Active-Active
 - Data types
 - Hashes
 - HyperLogLog
 - JSON
 - Lists
 - Sets
 - Sorted sets
 - Streams
 - Strings and bitfields
 - App failover
- Auto Tiering
 - Quick start
 - Manage storage engine
- Durability and availability
 - Clustering
 - Consistency
 - Discovery service
 - Replication
- Memory and performance
 - Eviction policy
 - Memory limits
 - Shard placement policy
- Clusters
 - Set up cluster
 - Add a node
 - Configure
 - Cluster settings
 - License keys
 - Sync node clocks
 - Rack-zone awareness
 - Optimize
 - Benchmark
 - Disk sizing
 - Environment optimization
 - Free system memory
 - Redis OSS Cluster API
 - WAIT command
 - Maintenance mode
 - Recover a cluster
 - Remove node
 - Replace node
 - Logging
 - Log security
 - rsyslog
 - Cluster alerts/events
 - Database alerts/events
 - Node alerts/events
 - User events
 - Slow log
 - Monitoring
 - Prometheus integration
 - Prometheus metrics
 - Uptrace integration
 - Nagios integration
- Networking
 - AWS Route 53 DNS
 - Cluster load balancer setup
 - Configure cluster DNS
 - mDNS client prerequisites
 - Multi-IP and IPv6
 - Network ports
 - Public and private endpoints
- Security
 - Access control
 - Cluster management access
 - Database access
 - Manage users

- Add users
- Manage user login and session
- Manage default user
- Manage passwords
 - Password complexity rules
 - Password expiration
 - Rotate passwords
 - Update Active-Active admin credentials
- Role-based access control
 - Configure ACLs
 - Create roles
 - Assign roles to users
- LDAP authentication
 - Enable role-based LDAP
 - Map LDAP groups to roles
 - Update database ACLs
 - Migrate to role-based LDAP
- Audit events
- Internode encryption
- Admin console security
 - Encrypt API requests
- Certificates
 - Create certificates
 - Update certificates
 - Enable OCSP stapling
- TLS
 - Enable TLS
 - Configure TLS protocol
 - Configure cipher suites
- Reference
 - Redis clients
 - Benchmark Auto Tiering
 - Command-line utilities
 - crdb-cli (manage Active-Active)
 - crdb
 - add-instance
 - create
 - delete
 - flush
 - get
 - health-report
 - list
 - purge-instance
 - remove-instance
 - update
 - task
 - cancel
 - list
 - status
 - redis-cli (run Redis commands)
 - rladmin (manage cluster)
 - bind
 - cluster
 - certificate
 - config
 - create
 - debug_info
 - join
 - master
 - ocsp
 - recover
 - reset_password
 - running_actions
 - stats_archiver
 - failover
 - help
 - info
 - migrate
 - node
 - addr
 - enslave

- external_addr
 - maintenance_mode
 - recovery_path
 - remove
 - snapshot
 - placement
 - recover
 - restart
 - status
 - suffix
 - tune
 - upgrade
 - verify
 - rlcheck (verify nodes)
- Metrics
 - Auto Tiering
 - Database operations
 - Resource usage
 - Open source compatibility
 - Commands
 - Cluster management
 - Connection management
 - Data types
 - Keys (generic)
 - Pub/sub
 - Scripting
 - Server management
 - Transactions
 - Configuration settings
 - RESP
 - REST API
 - Quick start
 - Requests
 - actions
 - bdbs
 - actions
 - backup_reset_status
 - export
 - export_reset_status
 - import
 - import_reset_status
 - optimize_shards_placement
 - alerts
 - crdt_sources/alerts
 - modules
 - config
 - upgrade
 - passwords
 - peer_stats
 - replica_sources/alerts
 - stats
 - last
 - sync_source_stats
 - upgrade
 - bootstrap
 - validate
 - cluster
 - actions
 - alerts
 - auditing/db_conns
 - certificates
 - rotate
 - ldap
 - module-capabilities
 - policy
 - services_configuration
 - stats
 - last
 - update_cert
 - crdb_tasks
 - crdbs

- flush
 - health_report
 - purge
 - updates
 - endpoints/stats
 - jsonschema
 - ldap_mappings
 - license
 - logs
 - modules
 - config/bdb
 - upgrade/bdb
 - nodes
 - actions
 - alerts
 - stats
 - last
 - status
 - ocsp
 - status
 - test
 - redis_acls
 - roles
 - shards/stats
 - last
 - suffix
 - suffixes
 - users
 - authorize
 - password
 - refresh_jwt
- Objects
 - action
 - alert
 - bdb
 - backup_location/export_location
 - dataset_import_sources
 - replica_sources status
 - replica_sync
 - snapshot_policy
 - status
 - syncer_sources
 - bdb_group
 - bootstrap
 - cluster_identity
 - credentials
 - identity
 - limits
 - node_identity
 - paths
 - policy
 - check_result
 - cluster
 - alert_settings
 - cluster_alert_settings_with_threshold
 - cluster_settings
 - crdb
 - cluster_info
 - database_config
 - health_report
 - health_report_configuration
 - instance_info
 - modify_request
 - crdb_task
 - db_alerts_settings
 - bdb_alert_settings_with_threshold
 - db_conns_auditing_config
 - job_scheduler
 - backup_job_settings
 - cert_rotation_job_settings
 - log_rotation_job_settings

- [node_checks_job_settings](#)
- [redis_cleanup_job_settings](#)
- [rotate_ccs_job_settings](#)
- [jwt_authorize](#)
- [ldap](#)
- [ldap_mapping](#)
- [module](#)
- [node](#)
- [ocsp](#)
- [ocsp_status](#)
- [proxy](#)
- [redis_acl](#)
- [role](#)
- [services_configuration](#)
 - [cm_server](#)
 - [crdb_coordinator](#)
 - [crdb_worker](#)
 - [mdns_server](#)
 - [pdns_server](#)
 - [stats_archiver](#)
- [shard](#)
 - [backup](#)
 - [loading](#)
 - [sync](#)
- [state-machine](#)
- [statistics](#)
 - [cluster metrics](#)
 - [DB metrics](#)
 - [node metrics](#)
 - [shard metrics](#)
- [suffix](#)
- [user](#)
- [Permissions](#)
- [Terminology](#)
- [Release notes](#)
 - [7.2.4 releases](#)
 - [7.2.4-52 \(August 2023\)](#)
 - [6.4.2 releases](#)
 - [6.4.2-94 \(July 2023\)](#)
 - [6.4.2-81 \(June 2023\)](#)
 - [6.4.2-69 \(May 2023\)](#)
 - [6.4.2-61 \(April 2023\)](#)
 - [6.4.2-43 \(March 2023\)](#)
 - [6.4.2-30 \(February 2023\)](#)
 - [6.2.18 releases](#)
 - [6.2.18-70 \(January 2023\)](#)
 - [6.2.18-65 \(December 2022\)](#)
 - [6.2.18-58 \(November 2022\)](#)
 - [6.2.18-49 \(October 2022\)](#)
 - [6.2.18-43 \(September 2022\)](#)
 - [6.2.12 \(August 2022\)](#)
 - [6.2.10 \(February 2022\)](#)
 - [6.2.8 \(October 2021\)](#)
 - [6.2.4 \(August 2021\)](#)
 - [6.0.20 \(April 2021\)](#)
 - [6.0.12 \(January 2021\)](#)
 - [6.0.8 \(September 2020\)](#)
 - [6.0 \(May 2020\)](#)
- [Previous releases](#)
 - [5.6.0 \(April 2020\)](#)
 - [5.5 preview \(April 2019\)](#)
 - [5.4.14 \(February 2020\)](#)
 - [5.4.10 \(December 2019\)](#)
 - [5.4.6 \(July 2019\)](#)
 - [5.4.4 \(June 2019\)](#)
 - [5.4.2 \(April 2019\)](#)
 - [5.4 \(December 2018\)](#)
 - [5.2.2 \(August 2018\)](#)
 - [5.3 beta\(July 2018\)](#)
 - [5.2 \(June 2018\)](#)
 - [5.0.2 \(2018 March\)](#)

- [5.0 \(November 2017\)](#)
- [4.5 \(May 2017\)](#)
- [4.4 \(December 2016\)](#)
- [4.3.0-230 \(August 2, 2016\)](#)
- [4.2.1-30 \(October 18, 2015\)](#)
- [4.0.0-49 \(June 18, 2015\)](#)
- [0.99.5-24 \(February 15, 2015\)](#)
- [0.99.5-11 \(January 5, 2015\)](#)

Redis Enterprise Software

Redis Enterprise is a self-managed, enterprise-grade version of Redis.

With Redis Enterprise, you get many enterprise-grade capabilities, including:

- Linear scalability
- High availability, backups, and recovery
- Predictable performance
- 24/7 support

You can run Redis Enterprise Software in an on-premises data center or on your preferred cloud platform.

Get started

Build a small-scale cluster with the Redis Enterprise Software container image.

- [Linux quickstart](#)
- [Docker quickstart](#)
- [Get started with Active-Active](#)

Install & setup

[Install & set up](#) a Redis Enterprise Software cluster.

- [Networking](#)
- [Set up & configure a cluster](#)
- [Release notes](#)

Databases

Create and manage a [Redis database](#) on a cluster.

- [Create a Redis Enterprise Software database](#)
- [Configure database](#)
- [Create Active-Active database](#)
- [Edit Active-Active database](#)

Security

[Manage secure connections](#) to the cluster and databases.

- [Access control](#)
- [Users & roles](#)
- [Certificates](#)
- [TLS & Encryption](#)

Reference

Use command-line utilities and the REST API to manage the cluster and databases.

- [rladmin, crdb-cli, & other utilities](#)
- [REST API reference & examples](#)
- [Redis commands \(redis.io\)](#)

Related info

- [Redis Enterprise Cloud](#)
- [Open source Redis \(redis.io\)](#)
- [Redis Stack](#)
- [Glossary](#)

Install, set up, and upgrade Redis Enterprise Software

Quickstarts

If you want to try out Redis Enterprise Software, see the following quickstarts:

- [Redis Enterprise Software quickstart](#)
- [Docker quickstart for Redis Enterprise Software](#)

Install Redis Enterprise Software

To install Redis Enterprise Software on a [supported platform](#), you need to:

1. [Plan your deployment.](#)
2. [Prepare to install.](#)
3. [Perform the install.](#)

Depending on your needs, you may also want to [customize the installation](#).

Upgrade existing deployment

If you already installed Redis Enterprise Software, you can:

- [Upgrade a cluster](#)
- [Upgrade a database](#)
- [Upgrade an Active-Active database](#)

Uninstall Redis Enterprise Software

- [Uninstall existing deployment](#)

More info and options

More information is available to help with customization and related questions:

- [CentOS/RHEL firewall configuration](#)
- [Change socket file location](#)
- [Cluster DNS configuration](#)
- [Cluster load balancer setup](#)
- [File locations](#)
- [Linux swap space configuration](#)
- [mDNS client prerequisites](#)
- [User and group ownership](#)

Next steps

After you install Redis Enterprise Software and set up your cluster, you can:

- [Add users](#) to the cluster with specific permissions. To begin, start with [Access control](#).
- [Create databases](#) to use with your applications.

Updated: August 17, 2023

Redis Enterprise Software quickstarts

Try out Redis Enterprise Software using one of the following quickstarts:

- [Redis Enterprise Software quickstart](#)
- [Docker quickstart for Redis Enterprise Software](#)

Additional quickstart guides are available to help you:

- Set up a [Auto Tiering cluster](#) to optimize memory resources.
- Set up an [Active-Active cluster](#) to enable high availability.
- [Benchmark](#) Redis Enterprise Software performance.

Updated: August 17, 2023

Redis Enterprise Software quickstart

This guide helps you install Redis Enterprise Software on a Linux host to test its capabilities.

When finished, you'll have a simple cluster with a single node:

1. [Ensure port availability](#)
2. [Install Redis Enterprise Software](#)
3. [Set up a Redis Enterprise Software cluster](#)
4. [Create a new Redis database](#)
5. [Connect to your Redis database](#)



Note: This quickstart is designed for local testing only. For production environments, see the [install and setup](#) guide for deployment options and instructions.

Ensure port availability

If port 53 is in use, the installation fails. This can occur in default installations of Ubuntu 18.04 and 20.04 in which `systemd-resolved` (DNS server) is running.

To prevent this issue, change the system configuration to make this port available before installation.

1. Edit `/etc/systemd/resolved.conf`:

```
sudo vi /etc/systemd/resolved.conf
```

2. Add `DNSStubListener=no` as the last line in the file and save the file.

3. Rename the current `/etc/resolv.conf` file:

```
sudo mv /etc/resolv.conf /etc/resolv.conf.orig
```

4. Create a symbolic link for `/etc/resolv.conf`:

```
sudo ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

5. Restart the DNS service:

```
sudo service systemd-resolved restart
```

Install Redis Enterprise Software

To install Redis Enterprise Software:

1. Download the installation files from the [Redis Enterprise Download Center](#) and copy the download package to a machine with a Linux-based OS.



Note: You are required to create a free account to access the download center.

2. Extract the installation files:

```
tar vxf <downloaded tar file name>
```

3. Run the install.sh script in the current directory:

```
sudo ./install.sh -y
```

Set up a cluster

To set up your machine as a Redis Enterprise Software cluster:

1. In the web browser on the host machine, go to <https://localhost:8443/new> to see the new Redis Enterprise Software Cluster Manager UI.

To use the legacy UI for this quickstart instead, see the [6.4 version of the quickstarts](#).



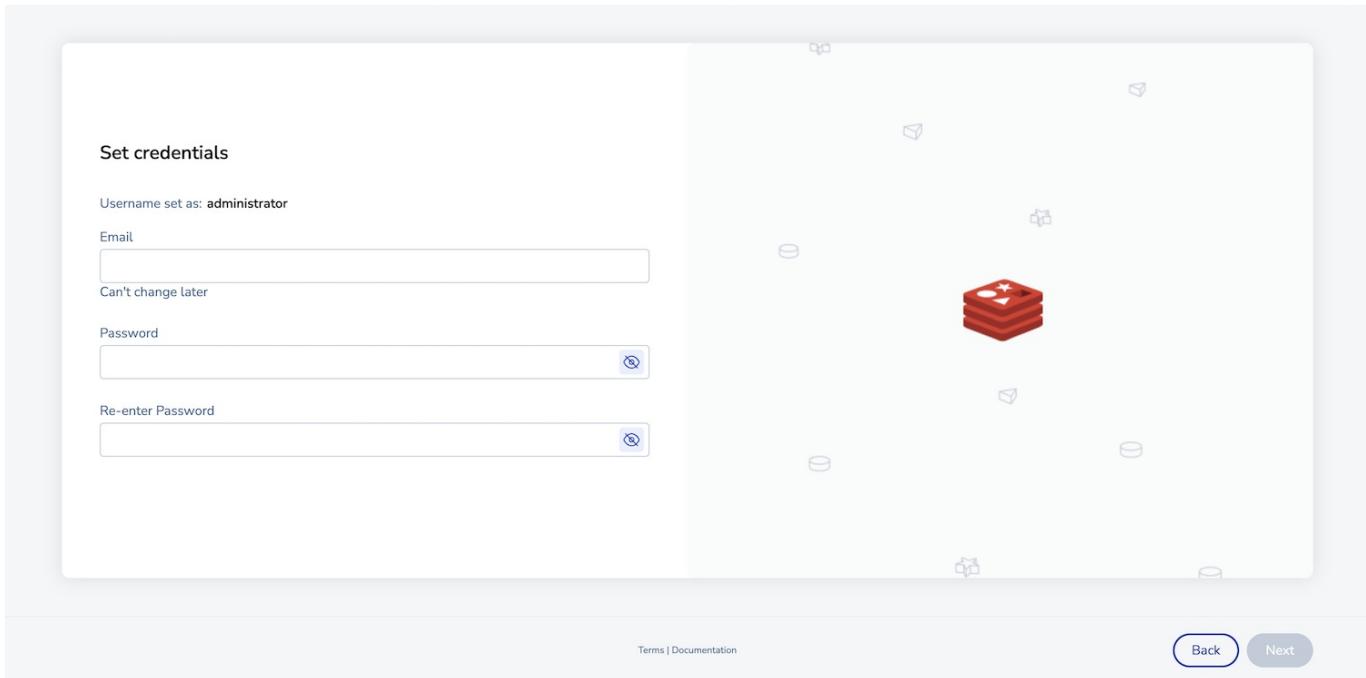
Note:

- If your browser displays a certificate error, you can safely proceed.
- If the server does not show the login screen, try again after a few minutes.

2. Select **Create new cluster**.



3. Enter an email and password for the administrator account, then select **Next** to proceed to cluster setup.



Set credentials

Username set as: administrator

Email

Can't change later

Password

Re-enter Password

Terms | Documentation

Back Next

You can also use these credentials to connect to the [REST API](#).

4. Enter your cluster license key if you have one. Otherwise, a trial version is installed.

License

Use professional plan key (optional)

Enter your unique cluster key here:

[Upload](#)

- (i)* You can upgrade to the professional plan at any point. Contact sales@redislabs.com to enjoy the full benefits of Redis Enterprise including autoscaling, rebalancing, automatic failover, and 24/7 support.

5. In the Configuration section, enter a cluster FQDN such as `cluster.local`, then select **Next**.

Configuration

Enable public endpoints support

When enabling public endpoints an additional internal FQDN will be created

FQDN (Fully Qualified Domain Name)

cluster.local

Rack zone awareness

When applicable, enables the cluster to place DB shards on nodes in different racks for high availability

6. On the node setup screen, select **Create cluster**.

7. Select **OK** to acknowledge the replacement of the HTTPS TLS certificate on the node. If you receive a browser warning, you can proceed safely.

Create a database

1. On the **Databases** screen, select **Quick database**.

The screenshot shows the redisenterprise web interface. The top navigation bar includes the logo, a back arrow, and the text "Databases". On the right, it shows "cluster.local" and "Administrator (Admin)". The left sidebar has links for Cluster, Nodes, Databases (which is selected and highlighted in blue), and Access Control. The main content area is titled "Databases" with a search bar and a "Filters" dropdown. A prominent "Create database" button is at the top right. A tooltip for "Two clicks DB" explains that it allows creating databases in seconds with the "Quick Database" button. Below this, a message says "You have no databases yet" and features a central icon of a cylinder with a plus sign. At the bottom, there are links for Support, Documentation, Terms, and a copyright notice: "© 2023 Redis".

2. Enter 12000 for the **Port**.

If port 12000 is not available, enter any available port number between 10000 to 19999 or leave it blank to let the cluster assign a port number for you. You will use this port number to connect to the database.

Quick Database

Cancel

Create

Database name

Database-24-Jul-23-18-40PM

Port

Will be set by the cluster
if not set manually

Memory limit (GB)

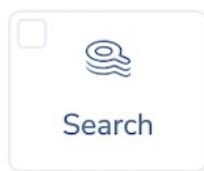
0.1



2.22 GB RAM available

Modules

Your selection cannot be changed at a later stage



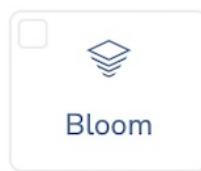
Search



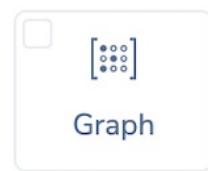
JSON



Timeseries

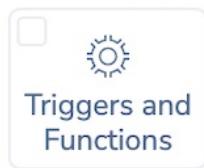


Bloom



Graph

Or



Triggers and
Functions

Modules parameters

3. Select **Create** to create your database.

When you see **Database active** appear on the database configuration screen, the database is activated and ready for you to use.

Database active

You now have a Redis database!

Connect to your database

After you create the Redis database, it is ready for you to store data. See the [test connectivity](#) page to learn how to connect to your database.

Supported web browsers

To use the Redis Enterprise Software admin console, you need a modern browser with JavaScript enabled.

The following browsers have been tested with the current version of the admin console:

- Microsoft Windows, version 10 or later.

- [Google Chrome](#), version 48 and later
 - [Microsoft Edge](#), version 20 and later
 - [Mozilla Firefox](#), version 44 and later
 - [Opera](#), version 35 and later
- Apple macOS:
 - [Google Chrome](#), version 48 and later
 - [Mozilla Firefox](#), version 44 and later
 - [Opera](#), version 35 and later
 - Linux:
 - [Google Chrome](#), version 49 and later
 - [Mozilla Firefox](#), version 44 and later
 - [Opera](#), version 35 and later

Next steps

- Connect to your Redis database with a [Redis client](#) and start adding data.
- Use the [memtier_benchmark quickstart](#) to check the cluster performance.

Updated: August 17, 2023

Docker quickstart for Redis Enterprise Software



Warning - Docker containers are currently only supported for development and test environments, not for production.

For testing purposes, you can run Redis Enterprise Software on Docker containers on Linux, Windows, or MacOS. The [Redis Enterprise Software container](#) acts as a node in a cluster.

To get started with a single Redis Enterprise Software container:

1. [Install Docker](#) for your operating system
2. [Run the Redis Enterprise Software Docker container](#)
3. [Set up a cluster](#)
4. [Create a new database](#)
5. [Connect to your database](#)

Deployment topologies

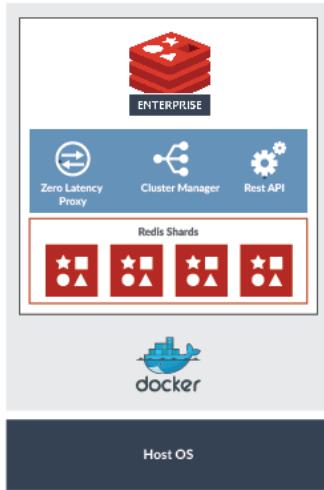
When deploying Redis Enterprise Software using Docker, several common topologies are available, according to your requirements:

- [Single-node cluster](#) – For local development or functional testing
- [Multi-node cluster on a single host](#) – For a small-scale deployment that is similar to production
- [Multi-node cluster with multiple hosts](#) – For more predictable performance or high availability compared to single-host deployments

Single node

The simplest topology is to run a single-node Redis Enterprise Software cluster with a single container on a single host machine. You can use this topology for local development or functional testing.

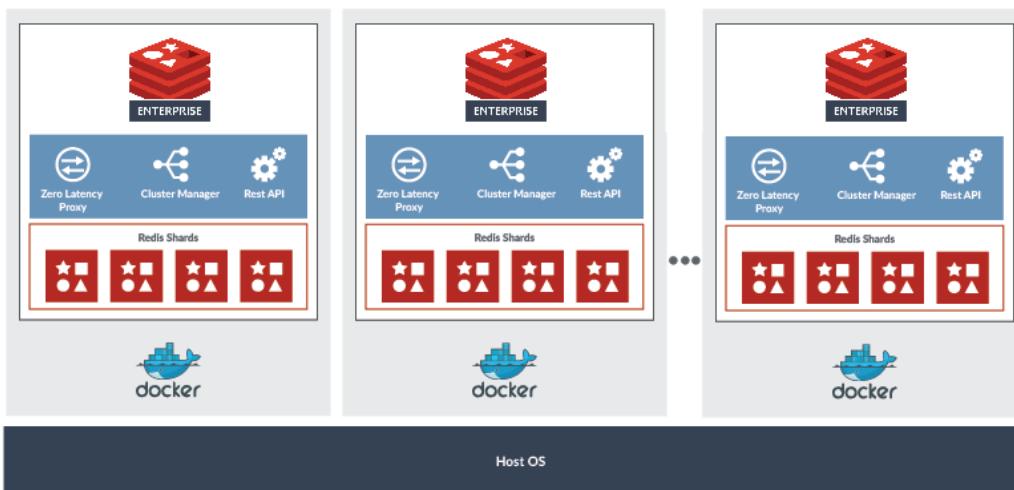
Single-node clusters have limited functionality. For example, Redis Enterprise Software can't use replication or protect against failures if the cluster has only one node.



Multiple nodes on one host

You can create a multi-node Redis Enterprise Software cluster by deploying multiple containers to a single host machine. The resulting cluster is scale minimized but similar to production deployments.

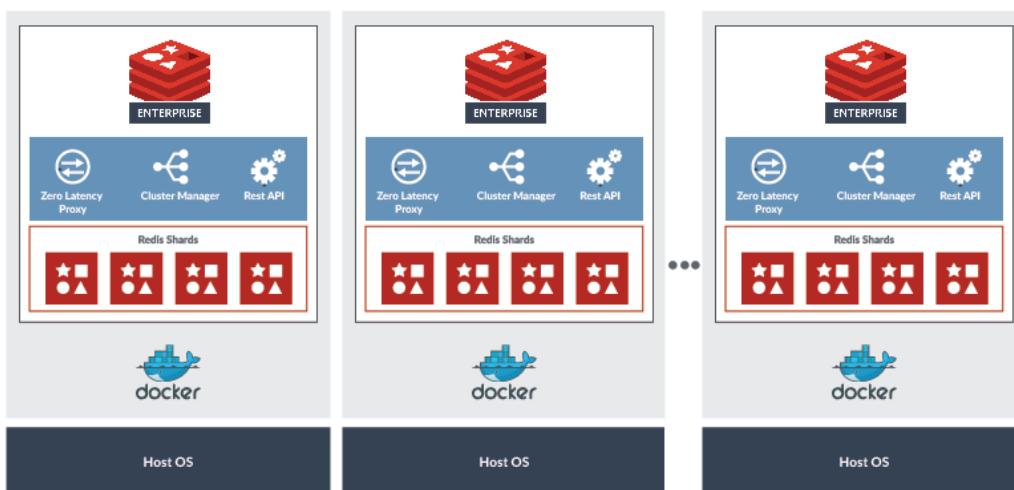
However, if you need predictable performance or high availability, don't host multiple nodes in containers on the same physical host.



Multiple nodes and hosts

You can also create a multi-node Redis Enterprise Software cluster with multiple containers by deploying each container to a different host machine.

This topology minimizes interference between containers, so Redis Enterprise Software performs more predictably than if you host multiple nodes on a single machine.



Install Docker

Follow the Docker installation instructions for your operating system:

- [Linux](#)
- [MacOS](#)
- [Windows](#)

Run the container

To download and start the Redis Enterprise Software Docker container, run the following `docker run` command in the terminal or command line for your operating system.



Note: On Windows, make sure Docker is configured to run Linux-based containers.

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

The example command runs the Docker container with Redis Enterprise Software on `localhost` and opens the following ports:

- Port 8443 for HTTPS connections
- Port 9443 for [REST API](#) connections
- Port 12000 for Redis client connections

You can publish other [ports](#) with `-p <host_port>:<container_port>` or use the `--network host` option to open all ports to the host network.

Set up a cluster

1. In the web browser on the host machine, go to <https://localhost:8443/new> to see the new Redis Enterprise Software Cluster Manager UI.

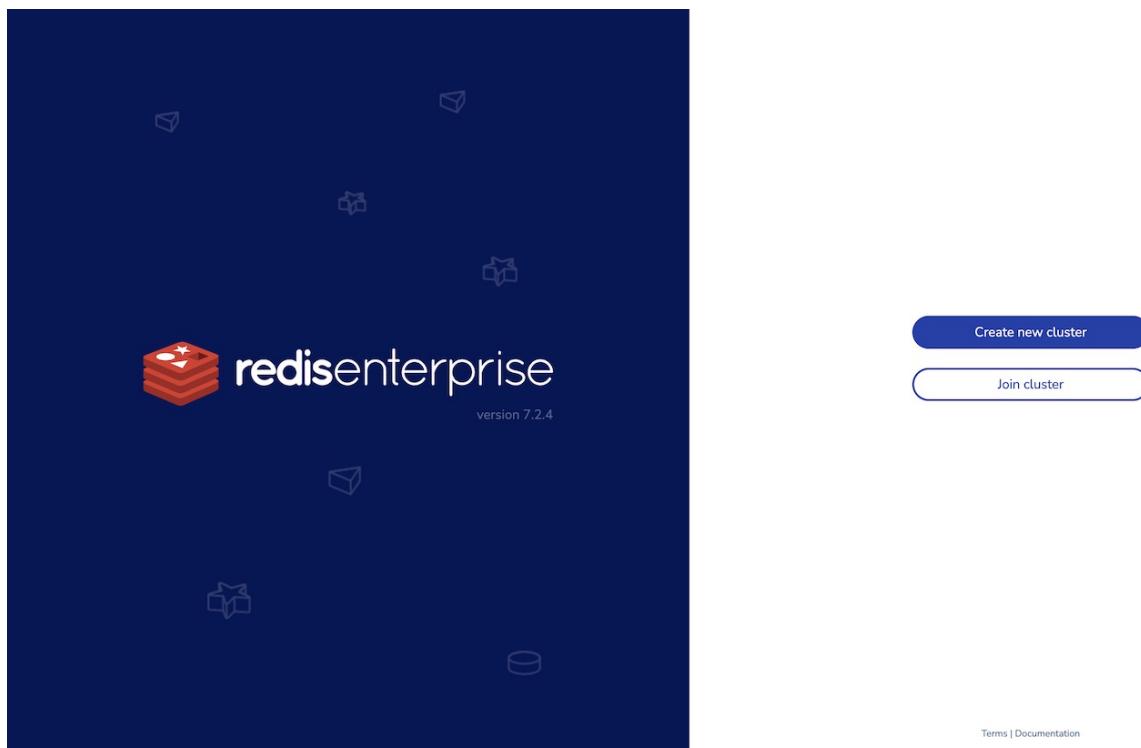
To use the legacy UI for this quickstart instead, see the [6.4 version of the quickstarts](#).



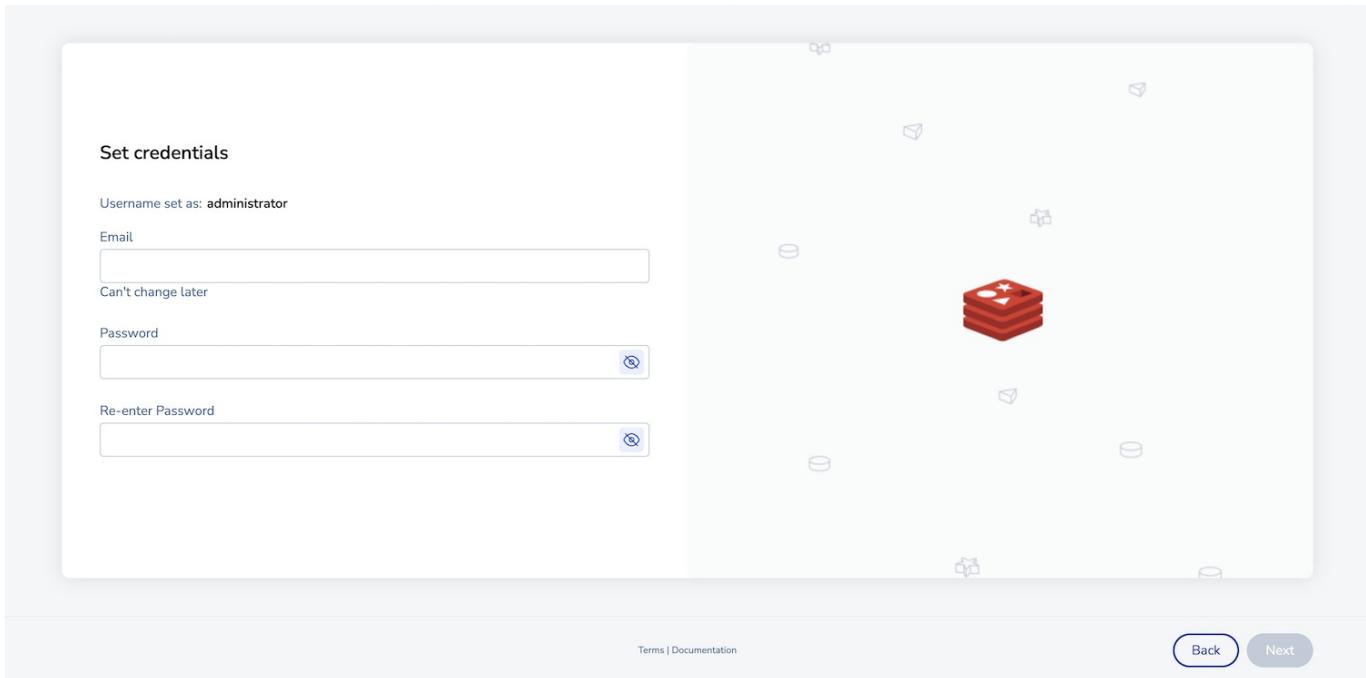
Note:

- If your browser displays a certificate error, you can safely proceed.
- If the server does not show the login screen, try again after a few minutes.

2. Select **Create new cluster**.



3. Enter an email and password for the administrator account, then select **Next** to proceed to cluster setup.



Set credentials

Username set as: administrator

Email

Can't change later

Password

Re-enter Password

Terms | Documentation

Back Next

You can also use these credentials to connect to the [REST API](#).

4. Enter your cluster license key if you have one. Otherwise, a trial version is installed.

License

Use professional plan key (optional)

Enter your unique cluster key here:

[Upload](#)

- (i)* You can upgrade to the professional plan at any point. Contact sales@redislabs.com to enjoy the full benefits of Redis Enterprise including autoscaling, rebalancing, automatic failover, and 24/7 support.

5. In the Configuration section, enter a cluster FQDN such as `cluster.local`, then select **Next**.

Configuration

Enable public endpoints support

When enabling public endpoints an additional internal FQDN will be created

FQDN (Fully Qualified Domain Name)

cluster.local

Rack zone awareness

When applicable, enables the cluster to place DB shards on nodes in different racks for high availability

6. On the node setup screen, select **Create cluster**.

7. Select **OK** to acknowledge the replacement of the HTTPS TLS certificate on the node. If you receive a browser warning, you can proceed safely.

Create a database

1. On the **Databases** screen, select **Quick database**.

The screenshot shows the redisenterprise web interface. The top navigation bar includes the logo, a back arrow, and the text "Databases". On the right, it shows "cluster.local" and "Administrator (Admin)". The left sidebar has links for Cluster, Nodes, Databases (which is selected and highlighted in blue), and Access Control. The main content area is titled "Databases" with a search bar and a "Filters" dropdown. A prominent "Create database" button is at the top right. A tooltip for "Two clicks DB" explains that it allows creating databases in seconds with the "Quick Database" button. Below this, a message says "You have no databases yet" and features a central icon of a cylinder with a plus sign. At the bottom, there are links for Support, Documentation, Terms, and a copyright notice: "© 2023 Redis".

2. Enter 12000 for the **Port**.

If port 12000 is not available, enter any available port number between 10000 to 19999 or leave it blank to let the cluster assign a port number for you. You will use this port number to connect to the database.

Quick Database

Cancel

Create

Database name

Database-24-Jul-23-18-40PM

Port

Will be set by the cluster
if not set manually

Memory limit (GB)

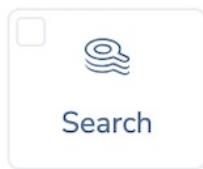
0.1



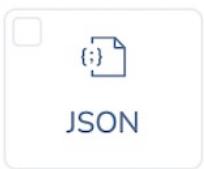
2.22 GB RAM available

Modules

Your selection cannot be changed at a later stage



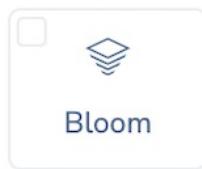
Search



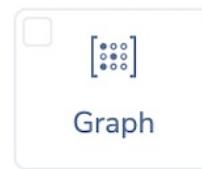
JSON



Timeseries

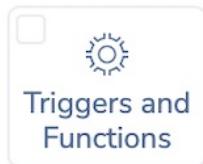


Bloom



Graph

Or



Triggers and
Functions

Modules parameters

3. Select **Create** to create your database.

When you see **Database active** appear on the database configuration screen, the database is activated and ready for you to use.

Database active

You now have a Redis database!



Note: If you cannot activate the database because of a memory limitation, make sure that Docker has enough memory allocated in the Docker Settings.

Connect to your database

After you create the Redis database, you can start storing data.

You can test connecting to your database with:

- [redis-cli](#)
- [Python application](#)

redis-cli

You can use the `redis-cli` command-line tool to interact with your Redis database.

1. Use `docker exec` to start an interactive shell session in the Redis Enterprise Software container:

```
docker exec -it rp bash
```

2. Run `redis-cli` and provide the port number with `-p` to connect to the database. Then use `SET` to store a key and `GET` to retrieve it.

```
$ /opt/redislabs/bin/redis-cli -p 12000
127.0.0.1:12000> SET key1 123
OK
127.0.0.1:12000> GET key1
"123"
```

Python

You can also run a Python application on the host machine to connect to your database.

 **Note:** The following section assumes you already have Python and the Redis Python client `redis-py` set up on the host machine running the container. For `redis-py` installation instructions, see the [Python client quickstart](#).

1. Create a new file called `redis_test.py` and add the following code:

```
import redis

r = redis.StrictRedis(host='localhost', port=12000, db=0)
print ("set key1 123")
print (r.set('key1', '123'))
print ("get key1")
print(r.get('key1'))
```

2. Run `redis_test.py` to store a key in your database and then retrieve it:

```
$ python redis_test.py
set key1 123
True
get key1
123
```

Next steps

- Connect to your Redis database with a [Redis client](#) and start adding data.
- Use the [memtier_benchmark quickstart](#) to check the cluster performance.

Updated: August 17, 2023

Install Redis Enterprise Software

After you [plan your deployment](#), [download a Redis Enterprise Software installation package](#), and finish [installation preparation](#):

1. [Install the Redis Enterprise Software package](#) on one of the nodes in the cluster.
2. Repeat this process for each node in the cluster.

For installation on machines without an internet connection, see [Offline installation](#).

Permissions and access

- Redis Enterprise Software installation creates the `redislabs:redislabs` user and group. Assigning other users to the `redislabs` group is optional. Users belonging to the `redislabs` group have permission to read and execute (e.g. use the `radmin status` command) but are not allowed to write (or delete) files or directories.
- Redis Enterprise Software is certified to run with permissions set to 750, an industry standard.



Warning - Do not reduce permissions to 700. This configuration has not been tested and is not supported.

More info and options

If you've already installed Redis Enterprise Software, you can also:

- [Upgrade an existing deployment.](#)
- [Uninstall an existing deployment.](#)

To learn more about customization and find answers to related questions, see:

- [CentOS/RHEL Firewall configuration](#)
- [Change socket file location](#)
- [Cluster DNS configuration](#)
- [Cluster load balancer setup](#)
- [File locations](#)
- [Supported platforms](#)
- [Manage installation questions](#)
- [mDNS client prerequisites](#)
- [User and group ownership](#)

Next steps

After your cluster is set up with nodes, you can:

- [Add users](#) to the cluster with specific permissions. To begin, start with [Access control](#).
- [Create databases](#) to use with your applications.

Updated: August 17, 2023

Plan Redis Enterprise Software deployment

Before installing Redis Enterprise Software, you need to:

- Set up your hardware. See [Hardware requirements](#) and [Persistent and ephemeral node storage](#) for more information.
- Choose your [deployment platform](#).

Redis Enterprise Software supports a variety of platforms, including:

- Multiple Linux distributions (Ubuntu, Red Hat Enterprise Linux (RHEL), IBM CentOS, Oracle Linux)
- [Amazon AWS AMI](#)
- [Docker container](#) (for development and testing only)
- [Kubernetes](#)

For more details, see [Supported platforms](#).

- Open appropriate [network ports](#) in the firewall to allow connections to the nodes.
- Configure [cluster DNS](#) so that cluster nodes can reach each other by DNS names.
- By default, the installation process requires an internet connection to install dependencies and synchronize the operating system clock. To learn more, see [Offline installation](#).

Next steps

After you finish planning your deployment, you can:

- Download an installation package.
- Prepare to install Redis Enterprise Software.
- View installation questions and prepare answers before installation.

Updated: August 17, 2023

Hardware requirements

The hardware requirements for Redis Enterprise Software are different for development and production environments.

- In a development environment, you can test your application with a live database.
If you want to test your application under production conditions, use the production environment requirements.
- In a production environment, you must have enough resources to handle the load on the database and recover from failures.

Development environment

You can build your development environment with non-production hardware, such as a laptop, desktop, or small VM or instance, and with these hardware requirements:

Item	Description	Minimum requirements	Recommended
Nodes per cluster	You can install on one node but many features require at least two nodes.	1 node	>= 2 nodes
RAM per node	The amount of RAM for each node.	4GB	>= 10GB
Storage per node	The amount of storage space for each node.	10GB	>= 20GB

Production environment

We recommend these hardware requirements for production systems or for development systems that are designed to demonstrate production use cases:

Item	Description	Minimum requirements	Recommended
Nodes* per cluster	At least three nodes are required to support a reliable, highly available deployment that handles process failure, node failure, and network split events in a consistent manner.	3 nodes	>= 3 nodes (Must be an odd number of nodes)
Cores* per node	Redis Enterprise Software is based on a multi-tenant architecture and can run multiple Redis processes (or shards) on the same core without significant performance degradation.	4 cores	>=8 cores
RAM* per node	Defining your RAM size must be part of the capacity planning for your Redis usage.	15GB	>=30GB
Ephemeral Storage	Used for storing replication files (RDB format) and cluster log files .	RAM x 2	>= RAM x 4
Persistent Storage	Used for storing snapshot (RDB format) and AOF files over a persistent storage media, such as AWS Elastic Block Storage (EBS) or Azure Data Disk.	RAM x 3	In-memory >= RAM x 6 (except for extreme 'write' scenarios) Auto Tiering >= (RAM + Flash) x 5.

Item	Description	Minimum requirements	Recommended
Network	We recommend using multiple NICs per node where each NIC is >100Mbps, but Redis Enterprise Software can also run over a single 1Gbps interface network used for processing application requests, inter-cluster communication, and storage access.	1G	>=10G

*Additional considerations:

- Nodes per Cluster:
 - Clusters with more than 35 nodes are not supported. Please contact the Redis support team for assistance if your sizing calls for deploying a larger number of nodes.
 - Quorum nodes also must comply with the above minimal hardware requirements.
 - To ensure synchronization and consistency, Active-Active deployments with three-node clusters should not use quorum nodes. Because quorum nodes do not store data shards, they cannot support replication. In case of a node failure, replica shards aren't available for Active-Active synchronization.
- Cores:
 - When the CPU load reaches a certain level, Redis Enterprise Software sends an alert to the operator.
 - If your application is designed to put a lot of load on your Redis database, make sure that you have at least one available core for each shard of your database.
 - If some of the cluster nodes are utilizing more than 80% of the CPU, consider migrating busy resources to less busy nodes.
 - If all the cluster nodes are utilizing over 80% of the CPU, consider scaling out the cluster by [adding a node](#).
- RAM:
 - Redis uses a relatively large number of buffers, which enable replica communication, client communication, pub/sub commands, and more. As a result, you should ensure that 30% of the RAM is available on each node at any given time.
 - If one or more cluster nodes utilizes more than 65% of the RAM, consider migrating resources to less active nodes.
 - If all cluster nodes are utilizing more than 70% of available RAM, consider [adding a node](#).
 - Do not run any other memory-intensive processes on the Redis Enterprise Software node.

Sizing considerations

General database sizing

Factors to consider when sizing your database.

- **Dataset size** – Your limit should be greater than your dataset size to leave room for overhead.
- **Database throughput** – High throughput needs more shards, leading to a higher memory limit.
- **Modules** – Using modules with your database consumes more memory.
- **Database clustering** – Allows you to spread your data into shards across multiple nodes.
- **Database replication** – Enabling replication doubles memory consumption.

Active-Active database sizing

Additional factors for sizing Active-Active databases:

- **Active-Active replication** – Requires double the memory of regular replication, which can be up to two times (2x) the original data size per instance.
- **Database replication backlog** – For synchronization between shards. By default, this is set to 1% of the database size.
- **Active-Active replication backlog** – For synchronization between clusters. By default, this is set to 1% of the database size.



Note: Active-Active databases have a lower threshold for activating the eviction policy, because it requires propagation to all participating clusters. The eviction policy starts to evict keys when one of the Active-Active instances reaches 80% of its memory limit.

Sizing databases with Auto Tiering enabled

Additional factors for sizing databases with Auto Tiering enabled:

- **Database persistence** – Auto Tiering uses dual database persistence where both the primary and replica shards persist to disk. This may add some processor and network overhead, especially in cloud configurations with network-attached storage.

Updated: August 17, 2023

Supported platforms

Redis Enterprise Software is supported on several operating systems, cloud environments, and virtual environments.

Supported platforms

ⓘ Supported – The platform is supported for this version of Redis Enterprise Software.

ⓘ Deprecated – The platform is still supported for this version of Redis Enterprise Software, but support will be removed in a future release.

ⓘ End of life – Platform support ended in this version of Redis Enterprise Software.

Redis Enterprise	7.2.4	6.4.2	6.2.18	6.2.12	6.2.10	6.2.8	6.2.4
Ubuntu¹							
20.04	ⓘ	ⓘ ⁶	–	–	–	–	–
18.04	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ
16.04	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ
RHEL & CentOS²							
8.8	ⓘ	–	–	–	–	–	–
8.7	ⓘ	ⓘ	–	–	–	–	–
8.5-8.6	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	–	–
8.0-8.4	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	–
7.0-7.9	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ
Oracle Linux³							
8	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	–	–
7	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ
Rocky Linux³							
8	ⓘ	ⓘ	ⓘ	–	–	–	–
Amazon Linux							
2	ⓘ	ⓘ ⁷	–	–	–	–	–
1	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ	ⓘ
Docker⁴							
Kubernetes⁵							

1. The server version of Ubuntu is recommended for production installations. The desktop version is only recommended for development deployments.

2. RHEL and CentOS deployments require OpenSSL 1.0.2 and [firewall configuration](#).

3. Based on the corresponding RHEL version.

4. [Docker images](#) of Redis Enterprise Software are certified for development and testing only.

5. See the [Redis Enterprise for Kubernetes documentation](#).

6. Ubuntu 20.04 support was added in Redis Enterprise Software [6.4.2-43](#).

7. A release candidate for Amazon Linux 2 support was added in Redis Enterprise Software [6.4.2-61](#). Official support for Amazon Linux 2 was added in Redis Enterprise Software [6.4.2-69](#).

Operating system limitations

TLS 1.0 and TLS 1.1

Redis Enterprise Software version 6.2.8 removed support for TLS 1.0 and TLS 1.1 on Red Hat Enterprise Linux 8 (RHEL 8) because that operating system [does not enable support](#) for these versions by default.

Ubuntu 20 rejects SHA1 certificates

With Ubuntu 20.04, you cannot use the SHA1 hash algorithm because [OpenSSL's security level is set to 2 by default](#). As a result, the operating system rejects SHA1 certificates, even if you enable the `mtls_allow_weak_hashing` option.

To avoid issues with SHA1 certificates, replace them with new certificates that use SHA-256. Note that certificates provided by Redis Enterprise Software use SHA-256.

Upgrade RHEL when using modules

RHEL 7 clusters cannot be directly upgraded to RHEL 8 when hosting databases using modules. Due to binary differences in modules between the two operating systems, you cannot directly update RHEL 7 clusters to RHEL 8 when those clusters host databases using modules. Instead, you need to create a new cluster on RHEL 8 and then migrate existing data from your RHEL 7 cluster. This does not apply to clusters that do not use modules.

This limitation is fixed for clusters using Redis Enterprise Software version 7.2 and later.

Modules not supported for Amazon Linux 2 release candidate

A database with modules cannot reside on an Amazon Linux 2 (release candidate) node. This limitation affects Redis Enterprise Software [6.4.2-61](#) but was fixed in version [6.4.2-69](#).

Modules cannot load in Oracle Linux 7 & 8

Databases hosted on Oracle Linux 7 & 8 cannot load modules.

As a temporary workaround, you can change the node's `os_name` in the Cluster Configuration Store (CCS):

```
ccs-cli hset node:<ID> os_name rhel7
```

Virtualization platforms

Redis Enterprise Software is compatible with VMware and other similar virtualization platforms. Make sure to do the following:

- Configure your memory, CPU, network, and storage settings to allow for optimal Redis Enterprise performance.
- Pin each Redis Enterprise shard to a specific ESX or ESXi host by setting the appropriate affinity rules.
- If you must manually migrate a virtual machine to another host, follow the best practices for shard maintenance and contact support if you have questions.
- Turn off VMware VMotion because Redis Enterprise is not compatible with VMotion.
- Don't use snapshots because Redis Enterprise cluster manages states dynamically, so a snapshot might not have the correct node and cluster states.

Updated: August 17, 2023

Persistent and ephemeral node storage

For each node in the cluster, you can configure paths for both persistent storage and ephemeral storage.



Note: The persistent storage and ephemeral storage discussed in this document are not related to Redis persistence or AWS ephemeral drives.

Persistent storage

Persistent storage is mandatory. The cluster uses persistent storage to store information that needs to persist even if a shard or a node fails, such as server logs, configurations, and files.

For example, if you configure [persistence](#) for a database, then the persistence information is stored in this location.

The persistent volume must be a storage area network (SAN) using an EXT4 or XFS file system and be connected as an external storage volume.

When using append-only file (AOF) persistence, use flash-based storage for the persistent volume.

Ephemeral storage

Ephemeral storage is optional. If configured, the cluster stores temporary information that does not need to persist in the ephemeral storage. This improves performance and helps reduce the load on the persistent storage.

Ephemeral storage must be a locally attached volume on each node.

Disk size requirements

For disk size requirements, see:

- [Hardware requirements](#) for general guidelines regarding the ideal disk size for each type of storage
- [Disk size requirements for extreme write scenarios](#) for special considerations when dealing with a high rate of write commands

Updated: August 17, 2023

File locations



Warning - To ensure that Redis Enterprise Software functions properly, be careful with the files in the application directories. If you modify or delete the application files, Redis Enterprise Software might not work as expected.

Application directories

The directories that Redis Enterprise Software installs into are:

Path	Description
/opt/redislabs	Main installation directory for all Redis Enterprise Software binaries
/opt/redislabs/bin	Binaries for all the utilities for command-line access and management, such as rladmin or redis-cli
/opt/redislabs/config	System configuration files
/opt/redislabs/lib	System library files
/opt/redislabs/sbin	System binaries for tweaking provisioning

Configuration and data directories

The default directories that Redis Enterprise Software uses for data and metadata are:

Path	Description
/var/opt/redislabs	Default storage location for the cluster data, system logs, backups, and ephemeral, persisted data
/var/opt/redislabs/log	System logs for Redis Enterprise Software
/var/opt/redislabs/run	Socket files for Redis Enterprise Software
/etc/opt/redislabs	Default location for cluster manager configuration and certificates
/tmp	Temporary files

You can change these file locations for:

- [Ephemeral and persistence storage](#) during cluster setup
- [Socket files](#) after cluster setup

Updated: August 17, 2023

Configure AWS EC2 instances for Redis Enterprise Software

There are some special considerations for installing and running Redis Enterprise Software on Amazon Elastic Cloud Compute (EC2) instances.

These include:

- Storage considerations
- Instance types
- Security group configuration

Storage considerations

AWS EC2 instances are ephemeral, but your persistent database storage should not be. If you require a persistent storage location for your database, the storage must be located outside of the instance. When you set up an instance, make sure it has a properly sized EBS-backed volume connected. When you set up Redis Enterprise Software on the instance, make sure that the [persistence storage](#) is configured to use this volume.



Note:

After [installing the Redis Enterprise Software package](#) on the instance and **before** running through [the setup process](#), you must give the group `redislabs` permission to the EBS volume by running the following command from the OS command-line interface (CLI):

```
chown redislabs:redislabs /< ebs folder name>
```

Another feature that may be of importance to you is the use of Provisioned IOPS for EBS-backed volumes. Provisioned IOPS guarantee a certain level of disk performance. There are two features in Redis Enterprise Software where this feature could be critical to use:

1. When using [Redis on Flash](#)
2. When using AOF on every write and there is a high write load. In this case, the provisioned IOPS should be on the nodes used as replicas in the cluster.

Instance types

Choose an instance type that has (at minimum) enough free memory and disk space to meet the Redis Enterprise Software [hardware requirements](#).

In addition, some instance types are optimized for EBS-backed volumes and some are not. If you are using persistent storage, you should use an instance type that is, especially if disk drain rate matters to your database implementation.

Security group configuration

When configuring the security group:

- Define a custom TCP rule for port 8443 to allow web browser access to the Redis Enterprise Software admin console from the IP address range you use to access the admin console.
- If you are using the DNS resolving option with Redis Enterprise Software, define a DNS UDP rule for port 53 to allow access to the databases' endpoints by using the [DNS resolving mechanism](#).
- To create a cluster that has multiple nodes all running as instances on AWS, you need to define a security group that has an All TCP rule for all ports, 0 - 65535, and add it to all instances that are part of the cluster. This ensures that all nodes are able to communicate with each other. To limit the number of open ports, you can open only the [ports used by Redis Enterprise Software](#).

After successfully launching the instances:

1. Install Redis Enterprise Software from the [Linux package or AWS AMI](#).
2. [Set up the cluster](#).

Updated: August 17, 2023

Prepare to install Redis Enterprise Software

Before you install Redis Enterprise Software:

- [Download an installation package](#).
- [View installation questions](#) and optionally prepare answers before installation.

- Review the [security considerations](#) for your deployment.
- Check that you have root-level access to each node, either directly or with sudo.
- Check that all [required ports are available](#).
- [Turn off Linux swap](#) on all cluster nodes.
- If you require the `redislabs` UID (user ID) and GID (group ID) numbers to be the same on all the nodes, create the `redislabs` user and group with the required numbers on each node.
- If you want to use Auto Tiering for your databases, see [Prepare Auto Tiering](#).

Next steps

- View [installation script options](#) before starting the installation.
- [Install Redis Enterprise Software](#).

Updated: August 17, 2023

Download a Redis Enterprise Software installation package

To download the installation package for any of the supported platforms:

1. Go to the [Redis download page](#).
2. Sign in with your Redis credentials or create a new account.
3. In the **Downloads** section for Redis Enterprise Software, select the installation package for your platform then select **Go**.



Note: Before you install the Linux package or AWS AMI on an AWS EC2 instance, review the [configuration requirements for AWS EC2 instances](#).

Updated: August 17, 2023

Ensure port availability

Before [installing Redis Enterprise Software](#), make sure all required ports are available.

Database ports

Make sure that the ports [Redis assigns to databases](#) are available and are not being used by the operating system or other processes.

To avoid port collision, update `/etc/sysctl.conf` to include:

```
net.ipv4.ip_local_port_range = 30000 65535
```

Port 53

If port 53 is in use, the installation fails. This can occur in default installations of Ubuntu 18.04 and 20.04 in which `systemd-resolved` (DNS server) is running.

To prevent this issue, change the system configuration to make this port available before installation.

1. Edit `/etc/systemd/resolved.conf`:

```
sudo vi /etc/systemd/resolved.conf
```

2. Add `DNSStubListener=no` as the last line in the file and save the file.

3. Rename the current `/etc/resolv.conf` file:

```
sudo mv /etc/resolv.conf /etc/resolv.conf.orig
```

4. Create a symbolic link for `/etc/resolv.conf`:

```
sudo ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

5. Restart the DNS service:

```
sudo service systemd-resolved restart
```

Updated: August 17, 2023

Prepare Auto Tiering

If you want to use Auto Tiering for your databases, review the prerequisites, storage requirements, and [other considerations](#) for RoF databases and prepare and format the flash memory.

Use the `prepare_flash` script to prepare and format flash memory:

```
sudo /opt/redislabs/sbin/prepare_flash.sh
```

This script finds unformatted disks and mounts them as RAID partitions in `/var/opt/redislabs/flash`.

To verify the disk configuration, run:

```
sudo lsblk
```

Updated: August 17, 2023

Install Redis Enterprise Software on Linux

After you [download a Redis Enterprise Software installation package](#), install it on one of the nodes in the cluster.

For installation on machines without an internet connection, see [Offline installation](#).

Install on Linux

To install Redis Enterprise Software using the command line:

1. Copy the installation package to the node.

(Optional) Use the [GPG key file](#) to confirm authenticity of Ubuntu/Debian or RHEL RPM packages:

- For Ubuntu:

1. Import the key:

```
gpg --import <path to GPG key>
```

2. Verify the package signature:

```
dpkg-sig --verify <path to installation package>
```

- For RHEL:

1. Import the key:

```
rpm --import <path to GPG key>
```

2. Verify the package signature:

```
rpm --checksig <path to installation package>
```

2. On the node, change to the directory where the installation package is located and extract the installation files:

```
tar vxf <tarfile name>
```

3. To start the installation process, run the installation script. See [installation script options](#) for a list of command-line options you can add to the following command:

```
sudo ./install.sh
```



Note:

- The Redis Enterprise Software files are installed in the default [file locations](#).
- By default, Redis Enterprise Software runs on the OS as the `redislabs` user and `redislabs` group. If needed, you can [specify a different user and group](#) during the installation.
- You must either be the root user or use `sudo` to run the installation script.

4. Answer the [installation questions](#) when shown to complete the installation process.



Note:

To skip the installation questions, use one of the following methods:

- Run `./install.sh -y` to answer yes to all of the questions.
- Create an [answer file](#) to answer installation questions automatically.

5. When installation completes successfully, the output displays the admin console's IP address:

```
Summary:
```

```
-----  
ALL TESTS PASSED.  
2017-04-24 10:54:15 [!] Please logout and login again to make  
sure all environment changes are applied.  
2017-04-24 10:54:15 [!] Point your browser at the following  
URL to continue:  
2017-04-24 10:54:15 [!] https://<your\_ip\_here>:8443
```

6. Repeat this process for each node in the cluster.

More info and options

To learn more about customization and find answers to related questions, see:

- [CentOS/RHEL firewall configuration](#)
- [Change socket file location](#)
- [Cluster DNS configuration](#)
- [Cluster load balancer setup](#)
- [mDNS client prerequisites](#)
- [File locations](#)
- [Supported platforms](#)

Limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.

2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Next steps

1. [Create](#) or [join](#) an existing Redis Enterprise Software cluster.

2. [Create a database](#).

For geo-distributed Active-Active replication, create an [Active-Active](#) database.

3. [Add users](#) to the cluster with specific permissions. To begin, start with [Access control](#).

Updated: August 17, 2023

Installation script command-line options

Run `./install.sh --help` to view command-line options supported by the installation script.

The following options are supported:

Option	Description
<code>-y</code>	Automatically answers yes to all install prompts, accepting all default values See Manage install questions
<code>-c <answer file></code>	Specify answer file used to respond to install prompts See Manage install questions
<code>-s <socket dir></code>	Specify directory for redislabs unix sockets (<i>new installs only</i>)
<code>--install-dir <dir></code>	Specifies installation directory (<i>new installs only</i>) See Customize install locations
<code>--config-dir <dir></code>	Configuration file directory (<i>new installs only</i>) See Customize install locations
<code>--var-dir <dir></code>	Var directory used for installation (<i>new installs only</i>) See Customize install locations
<code>--os-user <user></code>	Operating system user account associated with install; default: redislabs See Customize user and group (<i>new installs only</i>)
<code>--os-group <group></code>	Operating system group associated with install; default: redislabs See Customize user and group (<i>new installs only</i>)

Updated: August 17, 2023

Manage installation questions

Several questions are displayed during the Redis Enterprise Software installation process.

Here, you'll find a list of these questions and learn how to automatically answer these questions to perform a silent install.

Installation questions

Several questions appear during installation:

- **Linux swap file - Swap is enabled. Do you want to proceed? [Y/N]?**

We recommend that you [disable Linux swap](#) in the operating system configuration to give Redis Enterprise Software control of the memory allocation.

- **Automatic OS tuning - Do you want to automatically tune the system for best performance [Y/N]?**

To allow the installation process to optimize the OS for Redis Enterprise Software, answer Y. The installation process prompts you for additional information.

The /opt/redislabs/sbin/systune.sh file contains details about the tuning process.

- **Network time** - Do you want to set up NTP time synchronization now [Y/N]?

Redis Enterprise Software requires that all cluster nodes have synchronized time. You can either let the installation process configure NTP or you can [configure NTP manually](#).

- **Firewall ports** - Would you like to open RedisLabs cluster ports on the default firewall zone [Y/N]?

Redis Enterprise Software requires that all nodes have [specific network ports](#) open. To open the ports, you can:

- Answer Y to let the installation process open these ports.
- Answer N and configure the firewall manually for [RHEL/CentOS firewall](#).
- Answer N and configure the firewall on the node manually for your OS.

- **Installation verification (rlcheck)** - Would you like to run rlcheck to verify proper configuration? [Y/N]?

Run the rlcheck installation verification to make sure that the installation completed successfully. If you want to run this verification at a later time, you can run:

```
/opt/redislabs/bin/rlcheck
```

Answer install questions automatically

To perform a silent (or automated) install, answer the questions when you start the [install](#).

Answer yes to all questions

To automatically answer yes to all questions (which accepts the default values), run the [installation script](#) with the -y parameter:

```
./install.sh -y
```

Configure file to answer

Use an answer file to manage your response:

1. Create a text file to serve as an answer file.

The answer file can contain any of the parameters for the installation questions and indicate the answer for each question with yes or no.

For example:

```
ignore_swap=no
systune=yes
ntp=no
firewall=no
rlcheck=yes
```

If you use systune=yes, the installation answers yes to all of the system tuning questions.

2. Run the [installation script](#) with the -c command-line option and add the path to the answer file.

For example:

```
./install.sh -c /home/user/answers
```

Updated: August 17, 2023

Customize installation directories

When installing Redis Enterprise Software, you can customize the installation directories.

The files are installed in the redislabs directory located in the path that you specify.



Note:

- Custom installation directories are supported on Red Hat Enterprise Linux versions 7 and 8.
- When you install with custom directories, the installation does not run as an RPM file.
- If a `redislabs` directory already exists in the path that you specify, the installation fails.
- All nodes in a cluster must be installed with the same file locations.
- Custom installation directories are not supported for databases using Auto Tiering.

You can specify these file locations:

Files	Installer flag	Example parameter	Example file location
Binaries files	<code>--install-dir</code>	<code>/opt</code>	<code>/opt/redislabs</code>
Configuration files	<code>--config-dir</code>	<code>/etc/opt</code>	<code>/etc/opt/redislabs</code>
Data and log files	<code>--var-dir</code>	<code>/var/opt</code>	<code>/var/opt/redislabs</code>

These files are not in the custom directories:

- OS files
 - `/etc/cron.d/redislabs`
 - `/etc/firewalld/services`
 - `/etc/firewalld/services/redislabs-clients.xml`
 - `/etc/firewalld/services/redislabs.xml`
 - `/etc/ld.so.conf.d/redislabs_ldconfig.conf.tpl`
 - `/etc/logrotate.d/redislabs`
 - `/etc/profile.d/redislabs_env.sh`
 - `/usr/lib/systemd/system/rlec_supervisor.service.tpl`
 - `/usr/share/selinux/mls/redislabs.pp`
 - `/usr/share/selinux/targeted/redislabs.pp`
- Installation reference files
 - `/etc/opt/redislabs/redislabs_custom_install_version`
 - `/etc/opt/redislabs/redislabs_env_config.sh`

To specify directories during `installation`, include installer flags as [command-line options](#) when you run the `install.sh` script. For example:

```
sudo ./install.sh --install-dir <path> --config-dir <path> --var-dir <path>
```

Limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use custom installation directories.

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Updated: August 17, 2023

Customize system user and group

By default, Redis Enterprise Software is installed with the user:group `redislabs:redislabs`.

During installation, you can specify the user and group that own all Redis Enterprise Software processes.

If you specify the user only, then installation is run with the primary group that the user belongs to.



Note:

- Custom installation user is supported on Red Hat Enterprise Linux.
- When you install with custom directories, the installation does not run as an RPM file.
- You must create the user and group before attempting to install Redis Software.
- You can specify an LDAP user as the installation user.

To customize the user or group during [installation](#), include the `--os-user` or `--os-group` [command-line options](#) when you run the `install.sh` script. For example:

```
sudo ./install.sh --os-user <user> --os-group <group>
```

Updated: August 17, 2023

Offline installation

By default, the installation process requires an internet connection to enable installing dependency packages and for [synchronizing the operating system clock](#) against an NTP server.

If you install Redis Enterprise Software on a machine without an internet connection, you need to perform two tasks manually.

Install required dependency packages

When you install Redis Enterprise Software on a machine that is not connected to the internet, the installation process fails and displays an error message informing you it failed to automatically install dependencies. Review the installation steps in the console to see which missing dependencies the process attempted to install. Install all these dependency packages and then run the installation again.

Set up NTP time synchronization

At the end of the installation, the process asks if you want to set up NTP time synchronization. If you choose Yes while you are not connected to the internet, the action fails and displays the appropriate error message, but the installation completes successfully. Despite the successful completion of the installation, you still have to configure all nodes for [NTP time synchronization](#).

Updated: August 17, 2023

Upgrade an existing Redis Enterprise Software deployment

To upgrade Redis Enterprise Software:

1. Verify appropriate [network ports](#) are either open or used by Redis Enterprise Software.
2. [Upgrade the software on all nodes of the cluster](#).
3. [\(Optional\) Upgrade each database](#) in the cluster or [upgrade an Active-Active database](#) to enable new features and important fixes.

Updated: August 17, 2023

Upgrade a Redis Enterprise Software cluster

Supported upgrade paths

The following upgrade paths are supported:

Current cluster version	Upgrade to cluster version
6.4.x	7.2
6.2.x	7.2 6.4.x
6.0.x	7.2 6.4.x 6.2.x

Upgrade prerequisites

Before upgrading a cluster:

- Verify that you meet the upgrade path requirements for your desired cluster version and review the relevant [release notes](#) for any preparation instructions.
- Upgrade the cluster's primary (master) node first. To identify the primary node, use one of the following methods:
 - Nodes** screen in the new admin console (only available for Redis Enterprise versions 7.2 and later)
 - `rladmin status nodes` command
 - `GET /nodes/status` REST API request

Upgrade cluster

Starting with the primary (master) node, follow these steps for every node in the cluster. To ensure cluster availability, upgrade each node separately.

- Verify node operation with the following commands:

```
$ rlcheck
$ rladmin status extra all
```

- Download the Redis Enterprise Software installation package to the machine running the node.

- Extract the installation package:

```
tar vxf <tarfile name>
```



Note: You cannot change the installation path or the user during the upgrade.

- Run the install command. See [Installation script options](#) for a list of command-line options you can add to the following command:

```
sudo ./install.sh
```

The installation script automatically recognizes the upgrade and responds accordingly.

The upgrade replaces all node processes, which might briefly interrupt any active connections.

- Verify the node was upgraded to the new version and is still operational:

```
$ rlcheck
$ rladmin status extra all
```

- Visit the admin console.

If the admin console was open in a web browser during the upgrade, refresh the browser to reload the console.

After all nodes are upgraded, the cluster is fully upgraded. Certain features introduced in the new version of Redis Enterprise Software only become available after upgrading the entire cluster.

After upgrading from version 6.0.x to 6.2.x, restart `cnm_exec` on each cluster node to enable more advanced state machine handling capabilities:

```
supervisorctl restart cnm_exec
```

Updated: August 17, 2023

Upgrade a Redis Enterprise Software database

Default Redis database versions

When you upgrade an existing database or create a new one, it uses the default Redis version (`default_redis_version`) unless you specify the database version explicitly with `redis_version` in the [REST API](#) or `radmin upgrade db`.

Redis Enterprise Software v6.x includes two Redis database versions: 6.0 and 6.2. As of version 7.2, Redis Enterprise Software includes three Redis database versions.

To view available Redis database versions:

- In the admin console, see [Redis database versions](#) on the **Cluster > Configuration** screen.
- Send a [GET /nodes REST API request](#) and see `supported_database_versions` in the response.

The default Redis database version differs between Redis Enterprise releases as follows:

Redis Enterprise	Bundled Redis DB versions	Default DB version (upgraded/new databases)
7.2	6.0, 6.2, 7.2	7.2
6.4.2	6.0, 6.2	6.2
6.2.x	6.0, 6.2	6.0

- v6.2.x: `default_redis_version` is 6.0.

Setting `redis_upgrade_policy` to `major` enforces this default. However, if you change `redis_upgrade_policy` to `latest`, this enforces 6.2 as the default.

The upgrade policy is only relevant for Redis Enterprise Software versions 6.2.4 through 6.2.18. For more information about upgrade policies, see the [6.2 version of this document](#).

- v6.4.2: `default_redis_version` is 6.2.

Both `major` and `latest` upgrade policies use this new default.

You can override the default version with `radmin tune cluster default_redis_version <version>`; however, this might limit future upgrade options.

- v7.2: `default_redis_version` is 7.2.

Both `major` and `latest` upgrade policies use this new default.

Upgrade prerequisites

Before upgrading a database:

- Review the relevant [release notes](#) for any preparation instructions.
- Verify that the database version meets the minimums specified earlier.

To determine the database version:

- Use the admin console to open the **Configuration** tab for the database and select  **About**.
- (Optional) Use the `radmin status extra all` command to display configuration details:

```
radmin status extra all
```

When the database compatibility version is outdated, OLD REDIS VERSION appears in the command output.

- Verify the cluster is fully upgraded and operational.

Use the admin console to display the **Configuration** tab for the cluster. The tab displays the cluster version information and the Redis database compatibility version.

- To avoid data loss during the upgrade, [back up your data](#).

You can [export the data](#) to an external location, [enable replication](#), or [enable persistence](#).

When choosing how to back up data, keep the following in mind:

- To reduce downtime when replication is enabled, a failover is performed before restarting the primary (master) database.
- When persistence is enabled without replication, the database is unavailable during restart because the data is restored from the persistence file. AOF persistence restoration is slower than snapshot restoration.

Upgrade database

To upgrade a database:

1. *(Optional)* Back up the database to minimize the risk of data loss.
2. Use `radmin` to upgrade the database. During the upgrade process, the database will restart without losing any data.

- To upgrade a database without modules:

```
radmin upgrade db <database name | database ID>
```

- If the database has modules enabled and new module versions are available in the cluster, run `radmin upgrade db` with additional parameters to upgrade the module versions when you upgrade the database. See [Upgrade modules](#) for more details.
- To upgrade the database to a version other than the default version, use the `redis_version` parameter:

```
radmin upgrade db <database name | database ID> redis_version <version>
```

3. Check the Redis database compatibility version for the database to confirm the upgrade.

To do so:

- Use the admin console to open the **Configuration** tab for the database and select **About**.
- Use `radmin status databases extra all` to display a list of the databases in your cluster and their current Redis database compatibility version:

```
radmin status databases extra all
```

Verify that the Redis version is set to the expected value.

Updated: August 17, 2023

Upgrade an Active-Active database

When you upgrade an [Active-Active \(CRDB\) database](#), you can also upgrade the CRDB protocol version and feature version.

CRDB protocol version guidelines

Starting with version 5.4.2, a new CRDB protocol version helps support Active-Active features.

The new CRDB protocol is backward compatible, which means v5.4.2 CRDB instances can understand write operations from instances using the earlier CRDB protocol.

After you upgrade the CRDB protocol on one instance, non-upgraded instances cannot receive write updates from the upgraded instance.

The upgraded instance receives updates from upgraded and non-upgraded instances.

When upgraded to the latest protocol version, upgraded instances automatically receive any missing write operations.

Follow these upgrade guidelines:

- Upgrade all instances of a specific CRDB within a reasonable time frame to avoid temporary inconsistencies between the instances.
- Make sure that you upgrade all instances of a specific CRDB before you do global operations on the CRDB, such as removing instances and adding new instances.
- As of v6.0.20, protocol version 0 is deprecated and support will be removed in a future version.
- To avoid upgrade failures, update all Active-Active databases to the latest protocol version before upgrading Redis Enterprise Software to v6.0.20 or later.

Feature version guidelines

Starting with version 5.6.0, a new feature version (also called a *feature set version*) helps support new Active-Active features.

When you update the feature version for an Active-Active database, the feature version is updated for all database instances.

Follow these upgrade guidelines:

- As of v6.0.20, feature version 0 is deprecated and support will be removed in a future version.
- To avoid upgrade failures, update all Active-Active databases to the latest protocol version before upgrading Redis Enterprise Software to v6.0.20 or later.

Upgrade Active-Active database instance

To upgrade an Active-Active database (CRDB) instance:

1. [Upgrade Redis Enterprise Software](#) on each node in the clusters where the Active-Active instances are located.
2. To see the status of your Active-Active instances, run:

```
rladmin status
```

The statuses of the Active-Active instances on the node can indicate:

- OLD REDIS VERSION
- OLD CRDB PROTOCOL VERSION
- OLD CRBD FEATURESET VERSION

SHARDS:	DB_ID	NAME	ID	NODE	ROLE	SLOTS	USED_MEMORY	STATUS
db:3	crdb-test		redis:2	node:1	master	0-16383	3.78MB	OK, OLD CRDB PROTOCOL VERSION
db:3	crdb-test		redis:3	node:2	slave	0-16383	3.75MB	OK, OLD CRDB PROTOCOL VERSION

3. To upgrade each Active-Active instance, including the Redis version and CRDB protocol version, run:

```
rladmin upgrade db <database_name | database_ID>
```

If the protocol version is old, read the warning message carefully and confirm.

```
[rladmin> upgrade db crdb-test
WARNING: You are about to update the internal protocol of the CRDB instance.
Changes to the data in this CRDB instance will ONLY be replicated to upgraded CRDB instances.
We recommend that you upgrade all other CRDB instances of this database AS SOON AS POSSIBLE.
Please confirm [Y/N]: ]
```

The Active-Active instance uses the new Redis version and CRDB protocol version.

Use the `keep_crdt_protocol_version` option to upgrade the database feature version without upgrading the CRDB protocol version.

If you use this option, make sure that you upgrade the CRDB protocol soon after with the `rladmin upgrade db` command.

You must upgrade the CRDB protocol before you update the CRDB feature set version.

4. If the feature set version is old, you must upgrade all of the Active-Active instances. Then, to update the feature set for each active-active database, run:

```
crdb-cli crdb update --crdb-guid <CRDB-GUID> --featureset-version yes
```

You can retrieve the <CRDB-GUID> with the following command:

```
crdb-cli crdb list
```

Look for the fully qualified domain name (CLUSTER-FQDN) of your cluster and use the associated GUID:

CRDB-GUID	NAME	REPL-ID	CLUSTER-FQDN
700140c5-478e-49d7-ad3c-64d517ddc486	aatest	1	aatest1.example.com
700140c5-478e-49d7-ad3c-64d517ddc486	aatest	2	aatest2.example.com

Updated: August 17, 2023

Uninstall Redis Enterprise Software

Use the `rl_uninstall` script to uninstall Redis Enterprise Software and remove its files. `rl_uninstall` also deletes all Redis data and configuration.

For each node in the cluster, run:

```
sudo ./rl_uninstall.sh
```

Updated: August 17, 2023

Additional configuration

This section describes additional configuration options for Redis Enterprise Software installation.

Configure CentOS/RHEL firewall

Configure firewall rules for Redis Enterprise Software on CentOS or Red Hat Enterprise Linux (RHEL).

Configure swap for Linux

Turn off Linux swap space.

Change socket file locations

Change socket file locations.

Updated: July 21, 2021

Configure CentOS/RHEL firewall

CentOS and Red Hat Enterprise Linux (RHEL) distributions use `firewalld` by default to manage the firewall and configure `iptables`. The default configuration assigns the network interfaces to the `public` zone and blocks all ports except port 22, which is used for `SSH`.

When you install Redis Enterprise Software on CentOS or RHEL, it automatically creates two `firewalld` system services:

- A service named `redislabs`, which includes all ports and protocols needed for communication between cluster nodes.
- A service named `redislabs-clients`, which includes the ports and protocols needed for external communication (outside of the cluster).

These services are defined but not allowed through the firewall by default. During Redis Enterprise Software installation, the `installer` prompts you to

confirm auto-configuration of a default (public) zone to allow the **redislabs** service.

Although automatic firewall configuration simplifies installation, your deployment might not be secure if you did not use other methods to secure the host machine's network, such as external firewall rules or security groups. You can use firewalld configuration tools such as **firewall-cmd** (command line) or **firewall-config** (UI) to create more specific firewall policies that allow these two services through the firewall, as necessary.



Note: If databases are created with non-standard [Redis Enterprise Software ports](#), you need to explicitly configure firewalld to make sure those ports are not blocked.

Updated: August 17, 2023

Configure swap for Linux

Linux operating systems use swap space, which is enabled by default, to help manage memory (pages) by copying pages from RAM to disk. Due to the way Redis Enterprise Software utilizes and manages memory, it is best to prevent OS swapping. For more details, see [memory limits](#). The recommendation is to turn off Linux swap completely in the OS.

When you install or build the OS on the machine intended to host your Redis Enterprise Software cluster, avoid configuring swap partitions if possible.

Turn off swap

To turn off swap in the OS of an existing server, VM, or instance, you must have sudo access or be a root user to run the following commands:

1. Turn off swap:

```
sudo swapoff -a
```

2. Comment out the swap partitions configured in the OS so swap remains off even after a reboot:

```
sudo sed -i.bak '/ swap / s/^.*$/#1/g' /etc/fstab
```

Updated: August 17, 2023

Change socket file locations

Default socket file locations

There are two default locations for the socket files in Redis Enterprise Software:

- `/tmp` - In clean installations of Redis Enterprise Software version earlier than 5.2.2
- `/var/opt/redislabs/run` - In clean installations of Redis Enterprise Software version 5.2.2 and later

Note: The default location was changed in case you run any maintenance procedures that delete the `/tmp` directory.

When you upgrade Redis Enterprise Software from an earlier version to 5.2.2 or later, the socket files are not moved to the new location by default. You need to either specify a custom location for the socket files during [installation](#) or use the [following procedure](#) after installation.

Change socket file locations

To change the location of the socket files:

1. On each node in the cluster, run:

```
sudo rlutil create_socket_path socket_path=/var/opt/redislabs/run
```

2. Identify the node with the `master` role by running the following command on any node in the cluster:

```
rladmin status nodes
```

3. On the master node, change the socket file location:

```
sudo rlutil set_socket_path socket_path=/var/opt/redislabs/run
```

4. To update the socket file location for all other nodes, restart Redis Enterprise Software on each node in the cluster, one at a time:

```
sudo service rlec_supervisor restart
```

5. Restart each database in the cluster to update the socket file location:

```
rladmin restart db <db name>
```



Warning - Restarting databases can cause interruptions in data traffic.

Updated: August 17, 2023

Redis Enterprise Software product lifecycle

The Redis Enterprise Software product lifecycle fully reflects the [subscription agreement](#). However, for any discrepancy between the two policies, the subscription agreement prevails.

Redis Enterprise modules follow the [modules lifecycle](#).

Release numbers

Redis uses a four-place numbering scheme to designate released versions of its products. The format is “Major1.Major2.Minor-Build”.

- Major sections of the version number represents fundamental changes and additions in capabilities to Redis Enterprise Software. The Major1 and Major2 part of the version number are incremented based on the size and scale of the changes in each release.
- The Minor section of the version number represents quality improvements, fixes to existing capabilities, and new capabilities which are typically minor, feature-flagged, or optional.
- Build number is incremented with any changes to the product. Build number is incremented with each build when any change is made to the binaries.

Redis Enterprise Software typically gets two major releases every year but the product shipping cycles may vary. Maintenance releases, typically available on the last minor release of the current major1.major2 release are typically made available on a monthly cadence, although cycles may vary.

End-of-life schedule

Beginning with Redis Enterprise Software release 6.4, end-of-life (EOL) for a given Major release occurs 18 months after the formal release of the subsequent Major. Maintenance will only be provided on the last minor release of the major1.major2 releases. This update to the EOL policy ensures customers have a lead time of at least 18 months to upgrade to the new release after it is available.

Version - Release date	End of Life (EOL)
7.2 – August 2023	-
6.4 – February 2023	February 28, 2025
6.2 – August 2021	August 31, 2024
6.0 – May 2020	May 31, 2022
5.6 – April 2020	October 31, 2021
5.4 – December 2018	December 31, 2020
5.2 – June 2018	December 31, 2019



Note:

- EOL of release 6.2 was reset to occur 18 months after the release of version 6.4, in accordance with the updated EOL policy.
- EOL of release 6.4 will only be determined after the next major version is released and will be set to occur 18 months after that date.

Updated: August 17, 2023

Create a support package

If you encounter any issues that you are not able to resolve yourself and need to [contact Redis support](#) for assistance, you can create a support package that gathers all essential information to help debug your issues.



Note: The process of creating the support package can take several minutes and generates load on the system.

Admin console method

To create a support package:

1. In the navigation menu, select **Support**.

The screenshot shows the Redis Admin Console interface. On the left, there's a navigation menu with options: Cluster, Nodes, Databases (which is selected), and Access Control. The main area displays a list of databases with a search bar and filters. One database entry is visible: "Database-24-Jul-23-18-40PM". Below the list are buttons for "Items per page" (set to 10) and a "Proceed" button. A modal window titled "Support" is open, containing three contact methods: "Open a support ticket [helpdesk](#)", "Email us: support@redis.com", and "Call us: [+1-866-577-3347](#)". At the bottom of the modal, there are buttons for "Support", "Create a support package (recommended)" (which is highlighted in blue), and "Proceed". The footer of the page includes links for "Terms" and "© 2023 Redis".

2. Select **Proceed**.
3. In the **Create support package** dialog, select **Run process**.
4. The package is created and downloaded by your browser.

Command-line method

If package creation fails with `internal error` or if you cannot access the UI, create a support package for the cluster from the command line on any node in the cluster using the `radmin cluster debug_info` command:

```
/opt/redislabs/bin/radmin cluster debug_info
```

- If `radmin cluster debug_info` fails for lack of space in the `/tmp` directory, you can:

1. Change the storage location where the support package is saved:

```
radmin cluster config debuginfo_path <path>
```

The `redislabs` user must have write access to the storage location on all cluster nodes.

2. On any node in the cluster, run:

```
radmin cluster debug_info
```

- If `radmin cluster debug_info` fails for another reason, you can create a support package for the cluster from the command line on each node in the cluster with the command:

```
/opt/redislabs/bin/debuginfo
```

Upload the tar archive to [Redis support](#). The path to the archive is shown in the command output.

Updated: August 17, 2023

Manage databases

You can manage your Redis Enterprise Software databases with several different tools:

- Admin console (the web-based user interface)
- Command-line tools ([radmin](#), [redis-cli](#), [crdb-cli](#))
- [REST API](#)

Create a Redis Enterprise Software database

Create a database with Redis Enterprise Software.

Configure database settings

Configure settings specific to each database.

Connect to a database

Learn how to connect your application to a Redis database hosted by Redis Enterprise Software and test your connection.

Import and export data

How to import, export, flush, and migrate your data.

Recover a failed database

Recover a database after the cluster fails or the database is corrupted.

Delete databases

Delete a database from the admin console.

Active-Active geo-distributed Redis

Overview of the Active-Active database in Redis Enterprise Software

Auto Tiering

Auto Tiering enables your data to span both RAM and dedicated flash memory.

Durability and high availability

Overview of Redis Enterprise durability features such as replication, clustering, and rack-zone awareness.

Memory and performance

Learn more about managing your memory and optimizing performance for your database.

Updated: August 31, 2022

Create a Redis Enterprise Software database

Redis Enterprise Software lets you create databases and distribute them across a cluster of nodes.

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.
 - For the new Cluster Manager UI, go to <https://<hostname>:8443/new>
 - For the legacy UI, go to <https://<hostname>:8443>
2. Use one of the following methods to create a new database:
 - [Quick database](#)
 - [Create database](#) with additional configuration
3. If you did not specify a port number for the database, you can find the port number in the **Endpoint** field in the **Databases > Configuration > General** section.
4. [Test client connectivity.](#)

 **Note:** For databases with Active-Active replication for geo-distributed locations, see [Create an Active-Active database](#). To create and manage Active-Active databases, use the legacy UI.

Quick database

To quickly create a database and skip additional configuration options during initial creation:

1. On the **Databases** screen, select **Quick database**.
2. Configure settings that are required for database creation but can be changed later:
 - Database name
 - Memory limit (GB)
3. Configure optional settings that can't be changed after database creation:
 - Endpoint port (set by the cluster if not set manually)
 - Modules to enable
4. Optionally select **Full options** to configure [additional settings](#).
5. Select **Create**.

Create database

To create a new database and configure additional settings:

1. Either click the + button next to **Databases** in the navigation menu, or go to the **Databases** screen and select **Create database**.
2. Enter a **Database name**.

- Maximum of 63 characters
- Only letters, numbers, or hyphens (-) are valid characters
- Must start and end with a letter or digit
- Case-sensitive

3. To configure additional database settings, expand each relevant section to make changes.

See [Configuration settings](#) for more information about each setting.

4. Select **Create**.

Updated: August 17, 2023

Configure database settings

You can manage your Redis Enterprise Software databases with several different tools:

- [Admin console](#) (the web-based user interface)
- Command-line tools:
 - `rladmin` for standalone database configuration
 - `crdb-cli` for Active-Active database configuration
 - `redis-cli` for open source Redis configuration
- [REST API](#)

Edit database settings

You can change the configuration of a Redis Enterprise Software database at any time.

To edit the configuration of a database using the admin console:

1. On the **Databases** screen, select the database you want to edit.
2. From the **Configuration** tab, select **Edit**.
3. Change any [configurable database settings](#).



Note: For [Active-Active database instances](#), most database settings only apply to the instance that you are editing. To manage Active-Active databases, use the legacy UI.

4. Select **Save**.

Configuration settings

- **Name** - The database name requirements are:
 - Maximum of 63 characters
 - Only letters, numbers, or hyphens (-) are valid characters
 - Must start and end with a letter or digit
 - Case-sensitive
- **Endpoint port number** - You can define the port number that clients use to connect to the database. Otherwise, a port is randomly selected.



Note: You cannot change the [port number](#) after the database is created.

- **Memory limit** - [Database memory limits](#) include all database replicas and shards, including replica shards in database replication and database shards

in database clustering.

If the total size of the database in the cluster reaches the memory limit, the data eviction policy for the database is enforced.



Note: If you create a database with Auto Tiering enabled, you also need to set the RAM-to-Flash ratio for this database. Minimum RAM is 10%. Maximum RAM is 50%.

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



Note: To use modules, add them when you create a new database. You can't add a module to an existing database.

To add a module to the database:

1. In the **Modules** section, select one or more modules.
2. To customize module configuration, select **Modules parameters** and enter the optional custom configuration.
3. Select **Done**.

High availability & durability

- **Replication** - We recommend you use intra-cluster replication to create replica shards for each database for high availability.

If the cluster is configured to support [rack-zone awareness](#), you can also enable rack-zone awareness for the database.

- **Replica high availability** - Automatically migrates replica shards to an available node if a replica node fails or is promoted to primary.
- **Persistence** - To protect against loss of data stored in RAM, you can enable data persistence and store a copy of the data on disk with snapshots or an Append Only File.
- **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.

Clustering

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

- Enable [database clustering](#) and select the number of database shards.

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](#), which is compatible with open source Redis, 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.
- Sharding
- [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.

If you enable the OSS Cluster API, the shards placement policy and database proxy policy automatically change to *Sparse* and *All primary shards*.

- **Shards placement policy** - Determines how to distribute database shards across nodes in the cluster.
 - *Dense* places shards on the smallest number of nodes.
 - *Sparse* spreads shards across many nodes.
- **Database proxy policy** - Determines the number and location of active proxies, which manage incoming database operation requests.

Replica Of

With [Replica Of](#), you can make the database a repository for keys from other databases.

Scheduled backup

You can configure [periodic backups](#) of the database, including the interval and backup location parameters.

Alerts

Select [alerts](#) to show in the database status and configure their thresholds.

You can also choose to [send alerts by email](#) to relevant users.

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.

Database access

- **Unauthenticated access** - You can access the database as the default user without providing credentials.
- **Password-only authentication** - When you configure a password for your database's default user, 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.

Creating a database without ACLs enables a *default* user with full access to the database. You can secure default user access by requiring a password.

- **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 for a database:

1. In **Security > Access Control > Access Control List**, select **+ Add ACL**.
2. Select a [role](#) to grant database access.
3. Associate a [Redis ACL](#) with the role and database.
4. Select the check mark to add the ACL.

Internode encryption

Enable [Internode encryption](#) to encrypt data in transit between nodes for this database. See [Internode encryption](#) for more information.

Updated: August 17, 2023

Enable 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.

Prerequisites

The Redis OSS Cluster API is supported only when a database meets specific criteria.

The database must:

- Use the standard [hashing policy](#).

- Have the [proxy policy](#) set to either all-master-shards or all-nodes.

In addition, the database must *not*:

- Use node include or exclude in the proxy policy.
- Use [RediSearch](#), [RedisTimeSeries](#), or [RedisGears](#) modules.

The OSS Cluster API setting applies to individual databases, as opposed to the overall cluster.

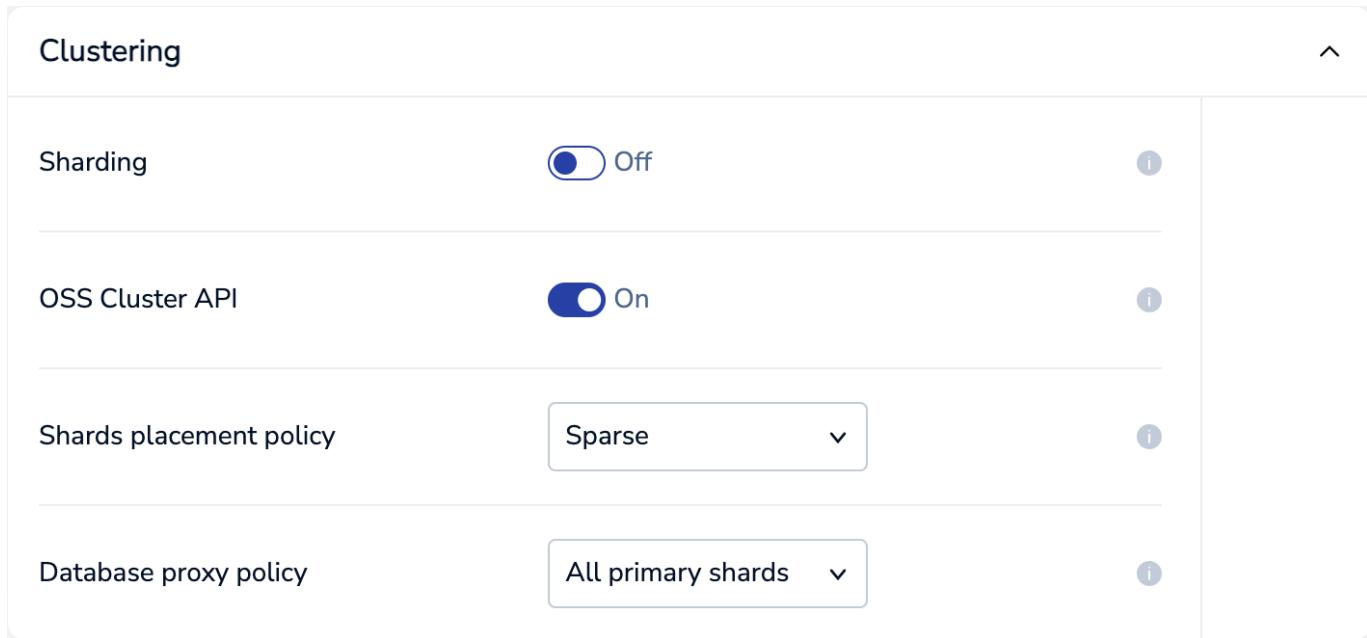
Enable OSS Cluster API support

You can use the admin console or the [rladmin](#) utility to enable OSS Cluster API support for a database.

Admin console

To enable the OSS Cluster API from the admin console for an existing database:

1. From the database's **Configuration** tab, select **Edit**.
2. Expand the **Clustering** section.
3. Turn on the **OSS Cluster API** toggle.



4. Select **Save**.

You can also use the admin console to enable the setting when creating a new database.

Command line ([rladmin](#))

You can use the [rladmin utility](#) to enable the OSS Cluster API for Redis Enterprise Software databases, including Replica Of (Active-Passive) databases.

For Active-Active (CRDB) databases, [use the crdb-cli utility](#).

To enable the OSS Cluster API for a Redis database from the command line:

```
$ rladmin tune db <database name or ID> oss_cluster enabled
```

To determine the current setting for a database from the command line, use `rladmin info db` to return the value of the `oss_cluster` setting.

```
$ rladmin info db test | grep oss_cluster:
oss_cluster: enabled
```

The Redis OSS Cluster API setting applies to the specified database only; it does not apply to the cluster.

Active-Active databases

Use the `crdb-cli` utility to enable the OSS Cluster API for Active-Active databases:

```
$ crdb-cli crdb update --crdb-guid <GUID> --oss-cluster true
```

For best results, you should do this when you first create the database.

Here's the basic process:

1. Create the Active-Active database:

```
$ crdb-cli crdb create --name <name> \
--memory-size 10g --port <port> \
--sharding true --shards-count 2 \
--replication true --oss-cluster true \
--instance fqdn=<fqdn>,username=<user>,password=<pass> \
--instance fqdn=<fqdn>,username=<user>,password=<pass> \
--instance fqdn=<fqdn>,username=<user>,password=<pass>
```

2. Obtain the CRDB-GUID ID for the new database:

```
$ crdb-cli crdb list
CRDB-GUID      NAME    REPL-ID   CLUSTER-FQDN
<CRDB-GUID>  Test      4        cluster1.local
```

3. Use the CRDB-GUID ID to enable the OSS Cluster API:

```
$ crdb-cli crdb update --crdb-guid <CRDB-GUID> \
--oss-cluster true
```

The Redis OSS Cluster API setting applies to all of the instances of the Active-Active database.

Turn off OSS Cluster API support

To deactivate OSS Cluster API support for a database, either:

- Use the admin console to turn off the **OSS Cluster API** toggle from the database **Configuration** settings.
- Use the appropriate utility to deactivate the OSS cluster setting.

For standard databases, including Replica Of (Active-Passive), use `rladmin`:

```
$ rladm tune db <Name or ID> oss_cluster disable
```

For Active-Active databases, use `crdb-cli`:

```
$ crdb-cli crdb update --crdb-guid <CRDB-GUID> \
--oss-cluster false
```

Multi-key command support

When you enable the Redis OSS Cluster API for a database, [multi-key commands](#) are only allowed when all keys are mapped to the same slot.

To verify that your database meets this requirement, make sure that the `CLUSTER KEYSLOT` reply is the same for all keys affected by the multi-key command. To learn more, see [multi-key operations](#).

Updated: August 17, 2023

Configure database persistence

All data is stored and managed exclusively in either RAM or RAM + flash Memory ([Redis on Flash](#)) and therefore, is at risk of being lost upon a process or

server failure. As Redis Enterprise Software is not just a caching solution, but also a full-fledged database, [persistence](#) to disk is critical. Therefore, Redis Enterprise Software supports persisting data to disk on a per-database basis and in multiple ways.

[Persistence](#) can be configured either during database creation or by editing an existing database's configuration. While the persistence model can be changed dynamically, it can take time for your database to switch from one persistence model to the other. It depends on what you are switching from and to, but also on the size of your database.

Configure database persistence

You can configure persistence when you [create a database](#), or you can edit an existing database's configuration:

1. From the **Databases** list, select the database, then select **Configuration**.
2. Select **Edit**.
3. Expand the **High Availability** section.
4. For **Persistence**, select an [option](#) from the list.
5. Select **Save**.

Data persistence options

There are six options for persistence in Redis Enterprise Software:

Options	Description
None	Data is not persisted to disk at all.
Append-only file (AOF) - fsync every write	Data is fsynced to disk with every write.
Append-only file (AOF) - fsync every 1 sec	Data is fsynced to disk every second.
Snapshot, every 1 hour	A snapshot of the database is created every hour.
Snapshot, every 6 hours	A snapshot of the database is created every 6 hours.
Snapshot, every 12 hours	A snapshot of the database is created every 12 hours.

Select a persistence strategy

When selecting your persistence strategy, you should take into account your tolerance for data loss and performance needs. There will always be tradeoffs between the two. The `fsync()` system call syncs data from file buffers to disk. You can configure how often Redis performs an `fsync()` to most effectively make tradeoffs between performance and durability for your use case. Redis supports three `fsync` policies: every write, every second, and disabled.

Redis also allows snapshots through RDB files for persistence. Within Redis Enterprise, you can configure both snapshots and `fsync` policies.

For any high availability needs, use replication to further reduce the risk of data loss.

For use cases where data loss has a high cost:

Append-only file (AOF) - `fsync` every write - Redis Enterprise sets the open-source Redis directive `appendfsyncalways`. With this policy, Redis will wait for the write and the `fsync` to complete prior to sending an acknowledgement to the client that the data has written. This introduces the performance overhead of the `fsync` in addition to the execution of the command. The `fsync` policy always favors durability over performance and should be used when there is a high cost for data loss.

For use cases where data loss is tolerable only limitedly:

Append-only file (AOF) - `fsync` every 1 sec - Redis will `fsync` any newly written data every second. This policy balances performance and durability and should be used when minimal data loss is acceptable in the event of a failure. This is the default Redis policy. This policy could result in between 1 and 2 seconds worth of data loss but on average this will be closer to one second.



Note: If you use AOF for persistence, enable replication to improve performance. When both features are enabled for a database, the replica handles persistence, which prevents any performance impact on the master.

For use cases where data loss is tolerable or recoverable for extended periods of time:

- Snapshot, every 1 hour - Performs a full backup every hour.
- Snapshot, every 6 hour - Performs a full backup every 6 hours.
- Snapshot, every 12 hour - Performs a full backup every 12 hours.
- None - Does not back up or persist data at all.

Append-only file (AOF) vs snapshot (RDB)

Now that you know the available options, to assist in making a decision on which option is right for your use case, here is a table about the two:

Append-only File (AOF)	Snapshot (RDB)
More resource intensive	Less resource intensive
Provides better durability (recover the latest point in time)	Less durable
Slower time to recover (Larger files)	Faster recovery time
More disk space required (files tend to grow large and require compaction)	Requires less resources (I/O once every several hours and no compaction required)

Active-Active data persistence

Active-Active databases support AOF persistence only. Snapshot persistence is not supported for Active-Active databases.

If an Active-Active database is using snapshot persistence, use `crdb-cli` to switch to AOF persistence:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> --default-db-config \
  '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

Auto Tiering data persistence

If you are enabling data persistence for databases running on Redis Enterprise Flash, by default both master and replica shards are configured to write to disk. This is unlike a standard Redis Enterprise Software database where only the replica shards persist to disk. This master and replica dual data persistence with replication is done to better protect the database against node failures. Flash-based databases are expected to hold larger datasets and repair times for shards can be longer under node failures. Having dual-persistence provides better protection against failures under these longer repair times.

However, dual data persistence with replication adds some processor and network overhead, especially for cloud configurations with network-attached persistent storage, such as EBS-backed volumes in AWS.

There may be times where performance is critical for your use case and you don't want to risk data persistence adding latency. If that is the case, you can disable data-persistence on the master shards using the following `radmin` command:

```
radmin tune db <database_ID_or_name> master_persistence disabled
```

Updated: August 17, 2023

Configure proxy policy

Redis Enterprise Software (RS) provides high-performance data access through a proxy process that manages and optimizes access to shards within the RS cluster. Each node contains a single proxy process. Each proxy can be active and take incoming traffic or it can be passive and wait for failovers.

Proxy policies

A database can have one of these proxy policies:

Proxy Policy	Description
Single	There is only a single proxy that is bound to the database. This is the default database configuration and preferable in most use cases.
All Master Shards	There are multiple proxies that are bound to the database, one on each node that hosts a database master shard. This mode fits most use cases that require multiple proxies.
All Nodes	There are multiple proxies that are bound to the database, one on each node in the cluster, regardless of whether or not there is a shard from this database on the node. This mode should be used only in special cases, such as using a load balancer .



Note: Manual intervention is also available via the rladm bind add and remove commands.

Database configuration

A database can be configured with a proxy policy using rladm bind.

Warning: Any configuration update which causes existing proxies to be unbounded can cause existing client connections to get disconnected.

You can run rladm to control and view the existing settings for proxy configuration.

The **info** command on cluster returns the existing proxy policy for sharded and non-sharded (single shard) databases.

```
$ rladm info cluster
cluster configuration:
  repl_diskless: enabled
  default_non_sharded_proxy_policy: single
  default_sharded_proxy_policy: single
  default_shards_placement: dense
  default_shards_overbooking: disabled
  default_fork_evict_ram: enabled
  default_redis_version: 3.2
  redis_migrate_node_threshold: 0KB (0 bytes)
  redis_migrate_node_threshold_percent: 8 (%)
  redis_provision_node_threshold: 0KB (0 bytes)
  redis_provision_node_threshold_percent: 12 (%)
  max_simultaneous_backups: 4
  watchdog profile: local-network
```

You can configure the proxy policy using the bind command in rladm. The following command is an example that changes the bind policy for a database called "db1" with an endpoint id "1:1" to "All Master Shards" proxy policy.

```
rladm bind db db1 endpoint 1:1 policy all-master-shards
```



Note: You can find the endpoint id for the endpoint argument by running `status` command for rladm. Look for the endpoint id information under the `ENDPOINT` section of the output.

Reapply policies after topology changes

If you want to reapply the policy after topology changes, such as node restarts, failovers and migrations, run this command to reset the policy:

```
rladm bind db <db_name> endpoint <endpoint id> policy <all-master-shards|all-nodes>
```

This is not required with single policies.

Other implications

During the regular operation of the cluster different actions might take place, such as automatic migration or automatic failover, which change what proxy needs to be bound to what database. When such actions take place the cluster attempts, as much as possible, to automatically change proxy bindings to adhere to the defined policies. That said, the cluster attempts to prevent any existing client connections from being disconnected, and hence might not entirely enforce the policies. In such cases, you can enforce the policy using the appropriate rladm commands.

About multiple active proxy support

RS allows multiple databases to be created. Each database gets an endpoint (a unique URL and port on the FQDN). This endpoint receives all the traffic for all operations for that database. By default, RS binds this database endpoint to one of the proxies on a single node in the cluster. This proxy becomes an active proxy and receives all the operations for the given database. (note that if the node with the active proxy fails, a new proxy on another node takes over as part of the failover process automatically).

In most cases, a single proxy can handle a large number of operations without consuming additional resources. However, under high load, network bandwidth or a high rate of packets per second (PPS) on the single active proxy can become a bottleneck to how fast database operation can be performed. In such cases, having multiple active proxies, across multiple nodes, mapped to the same external database endpoint, can significantly improve throughput.

With the multiple active proxies capability, RS enables you to configure a database to have multiple internal proxies in order to improve performance, in some cases. It is important to note that, even though multiple active proxies can help improve the throughput of database operations, configuring multiple active proxies may cause additional latency in operations as the shards and proxies are spread across multiple nodes in the cluster.



Note: When the network on a single active proxy becomes the bottleneck, you might also look into enabling the multiple NIC support in RS. With nodes that have multiple physical NICs (Network Interface Cards), you can configure RS to separate internal and external traffic onto independent physical NICs. For more details, refer to [Multi-IP & IPv6](#).

Having multiple proxies for a database can improve RS's ability for fast failover in case of proxy and/or node failure. With multiple proxies for a database, there is no need for a client to wait for the cluster to spin up another proxy and a DNS change in most cases, the client just uses the next IP in the list to connect to another proxy.

Updated: August 17, 2023

Configure high availability for replica shards

When you enable [database replication](#), Redis Enterprise Software copies your data to a replica node to make your data highly available. If the replica node fails or if the primary (master) node fails and the replica is promoted to primary, the remaining primary node is a single point of failure.

You can configure high availability for replica shards so that the cluster automatically migrates the replica shards to an available node. This process is known as *replica high availability* or *replica_ha*.

An available node:

1. Meets replica migration requirements, such as [rack-awareness](#).
2. Has enough available RAM to store the replica shard.
3. Does not also contain the master shard.

In practice, replica migration creates a new replica shard and copies the data from the master shard to the new replica shard.

For example:

1. Node:2 has a master shard and node:3 has the corresponding replica shard.
2. Either:
 - Node:2 fails and the replica shard on node:3 is promoted to master.
 - Node:3 fails and the master shard is no longer replicated to the replica shard on the failed node.
3. If replica HA is enabled, a new replica shard is created on an available node.
4. The data from the master shard is replicated to the new replica shard.



Note:

- Replica HA follows all prerequisites of replica migration, such as [rack-awareness](#).
- Replica HA migrates as many shards as possible based on available DRAM in the target node. When no DRAM is available, replica HA stops migrating replica shards to that node.

Configuring high availability for replica shards

Using `rladmin` or the REST API, replica HA is controlled on the database level and on the cluster level. You can enable or disable replica HA for a database or for the entire cluster.

When replica HA is enabled for both the cluster and a database, replica shards for that database are automatically migrated to another node in the event of a master or replica shard failure. If replica HA is disabled at the cluster level, replica HA will not migrate replica shards even if replica HA is enabled for a database.



Note: By default, replica HA is enabled for the cluster and disabled for each database.



Note: For Active-Active databases, replica HA is enabled for the database by default to make sure that replica shards are available for Active-Active replication.

To enable replica HA for a cluster using `rladmin`, run:

```
rladmin tune cluster slave_ha enabled
```

To disable replica HA for a specific database using rladmin, run:

```
rladmin tune db <bdb_uid> slave_ha disabled
```

Configuration options

You can see the current configuration options for replica HA with:

```
rladmin info cluster
```

Grace period

By default, replica HA has a 10-minute grace period after node failure and before new replica shards are created.

To configure this grace period from rladmin, run:

```
rladmin tune cluster slave_ha_grace_period <time_in_seconds>
```

Shard priority

Replica shard migration is based on priority. When memory resources are limited, the most important replica shards are migrated first:

1. **slave_ha_priority** - Replica shards with higher integer values are migrated before shards with lower values.

To assign priority to a database, run:

```
rladmin tune db <bdb_uid> slave_ha_priority <positive integer>
```

2. Active-Active databases - Active-Active database synchronization uses replica shards to synchronize between the replicas.
3. Database size - It is easier and more efficient to move replica shards of smaller databases.
4. Database UID - The replica shards of databases with a higher UID are moved first.

Cooldown periods

Both the cluster and the database have cooldown periods.

After node failure, the cluster cooldown period (**slave_ha_coldown_period**) prevents another replica migration due to another node failure for any database in the cluster until the cooldown period ends. The default is one hour.

After a database is migrated with replica HA, it cannot go through another migration due to another node failure until the cooldown period for the database (**slave_ha_bdb_coldown_period**) ends. The default is two hours.

To configure cooldown periods, use [rladmin tune cluster](#):

- For the cluster:

```
rladmin tune cluster slave_ha_coldown_period <time_in_seconds>
```

- For all databases in the cluster:

```
rladmin tune cluster slave_ha_bdb_coldown_period <time_in_seconds>
```

Alerts

The following alerts are sent during replica HA activation:

- Shard migration begins after the grace period.
- Shard migration fails because there is no available node (sent hourly).
- Shard migration is delayed because of the cooldown period.

Configure shard placement

In Redis Enterprise Software , the location of master and replica shards on the cluster nodes can impact the database and node performance. Master shards and their corresponding replica shards are always placed on separate nodes for data resiliency. The [shard placement policy](#) helps to maintain optimal performance and resiliency.

In addition to the shard placement policy, considerations that determine shard placement are:

- Separation of master and replica shards
- Available persistence and Auto Tiering storage
- [Rack awareness](#)
- Memory available to host the database when fully populated

The shard placement policies are:

- **dense** - Place as many shards as possible on the smallest number of nodes to reduce the latency between the proxy and the database shards; Recommended for Redis on RAM databases to optimize memory resources
- **sparse** - Spread the shards across as many nodes in the cluster as possible to spread the traffic across cluster nodes; Recommended for databases with Auto Tiering enabled to optimize disk resources

When you create a Redis Enterprise Software cluster, the default shard placement policy (dense) is assigned to all databases that you create on the cluster.

You can:

- Change the default shard placement policy for the cluster to **sparse** so that the cluster applies that policy to all databases that you create
- Change the shard placement policy for each database after the database is created

Default shard placement policy

When you create a new cluster, the cluster configuration has a dense default shard placement policy. When you create a database, this default policy is applied to the new database.

To see the current default shard placement policy, run `rladmin info cluster`:

```
bash-4.2$ rladmin info cluster
Cluster configuration:
  repl_diskless: enabled
  shards_overbooking: disabled
  default_non_sharded_proxy_policy: single
  default_sharded_proxy_policy: single
  default_shards_placement: sparse
```

To change the default shard placement policy so that new databases are created with the **sparse** shard placement policy, run:

```
rladmin tune cluster default_shards_placement [ dense | sparse ]
```

Shard placement policy for a database

To see the shard placement policy for a database in `rladmin status`.

DATABASES:							
DB-ID	NAME	TYPE	STATUS	SHARDS	PLACEMENT	REPLICATION	PERSISTENCE
db:1	db1	redis	active	2	sparse	disabled	disabled
db:2	db2	redis	active	2	dense	disabled	disabled

To change the shard placement policy for a database, run:

```
rladmin placement db [ database name | database ID ] [ dense | sparse ]
```

After you have [Set up a cluster](#) and [created a Redis database](#), you can connect to your database.

To connect to your database, you need the database endpoint, which includes the cluster name (FQDN) and the database port. Select the Configuration tab in the database screen to find the database endpoint.

If you try to connect with the FQDN, and the database does not respond, try connecting with the IP address. If this succeeds, DNS is not properly configured. To set up DNS, see [Configure cluster DNS](#).

If you want to secure your connection, set up [TLS](#).

Connect to a database

Use one of the following connection methods to connect to your database:

- [redis-cli](#) utility
- [Redis client](#) for your preferred programming language
- [RedisInsight](#)

redis-cli

The [redis-cli](#) utility is installed when you install Redis. It provides a command-line interface that lets you work with your database using core [Redis commands](#).

`redis-cli` is located in `/opt/redislabs/bin/` on a node with Redis Enterprise installed. To connect to the database from a node in the cluster, run `redis-cli` with the database port.

```
$ sudo /opt/redislabs/bin/redis-cli -p <port>
```

To connect using `redis-cli` from outside of the cluster, run `redis-cli` with the database hostname and port.

```
$ sudo /opt/redislabs/bin/redis-cli -h <host> -p <port>
```

Redis client

Different programming languages use different connection clients to interact with Redis databases, and each client has its own syntax and installation process. For help with a specific client, see the client's documentation.

See the [client list](#) to view all Redis clients by language.



Note:

You can't use client libraries to configure Redis Enterprise Software. Instead, use:

- The Redis Software [admin console](#)
- The [REST API](#)
- Command-line utilities, such as [rladmin](#)

Code example (Python)

A simple python application can also connect to the database.



Note: The following section assumes you already have Python and `redis-py` (python library for connecting to Redis) configured. You can find the instructions to configure `redis-py` on the [github page for redis-py](#).

1. Create a new file called `redis_test.py`:

```
import redis

r = redis.StrictRedis(host='<host>', port=<port>)
print ("set key1 123")
print (r.set('key1', '123'))
print ("get key1")
print(r.get('key1'))
```

Replace <host> and <port> with the hostname and port for your database.

2. Run the redis_test.py application to store and retrieve a key:

```
$ python redis_test.py
```

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

```
set key1 123
True
get key1
123
```

RedisInsight

RedisInsight is a free Redis GUI that is available for MacOS, Windows, and Linux.

1. [Install RedisInsight](#).
2. Open RedisInsight and select **Add Redis Database**.
3. Enter the host and port in the **Host** and **Port** fields.
4. Select **Use TLS** if **TLS** is set up.
5. Select **Add Redis Database** to connect to the database.

See the [RedisInsight documentation](#) for more information.

Updated: August 17, 2023

Supported connection clients

You can connect to Redis Enterprise Software databases programmatically using client libraries.

Redis client libraries

To connect an application to a Redis database hosted by Redis Enterprise Software, use a [client library](#) appropriate for your programming language.

You can also use the `redis-cli` utility to connect to a database from the command line.

For examples of each approach, see the [Redis Enterprise Software quickstart](#).

Note: You cannot use client libraries to configure Redis Enterprise Software. Instead, use:

- The Redis Software [admin console](#)
- The [REST API](#)
- Command-line utilities, such as `rladmin`

Discovery service

We recommend the following clients when using a [discovery service](#) based on the Redis Sentinel API:

- [redis-py](#) (Python Redis client)
- [Hiredis](#) (C Redis client)
- [Jedis](#) (Java Redis client)
- [node-redis](#) (Node.js Redis client)

If you need to use another client, you can use [Sentinel Tunnel](#) to discover the current Redis master with Sentinel and create a TCP tunnel between a local port on the client and the master.

Updated: August 17, 2023

Test client connection

In various scenarios, such as after creating a new cluster or upgrading the cluster, it is highly advisable to verify client connectivity to the database.

To test client connectivity:

1. [Create a Redis database](#) and get the database endpoint, which contains the cluster name (FQDN).
2. Try to connect to the database endpoint from your client of choice, and execute commands against the database.
3. If the database does not respond, try to connect to the database endpoint by using the IP address rather than the FQDN; if you succeed, it means that the DNS is not properly configured. For additional details, refer to [DNS](#).

If any issues are encountered during the connectivity test, contact our support at support@redislabs.com.

Test connecting to your database

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:

- Connecting with redis-cli, the built-in command-line tool
- Connecting with a *Hello World* application using Python.

Connecting using redis-cli

Run redis-cli, located in the /opt/redislabs/bin directory, to connect to port 12000 and store and retrieve a key in database1

```
# sudo /opt/redislabs/bin/redis-cli -p 12000
127.0.0.1:16653> set key1 123
OK
127.0.0.1:16653> get key1
"123"
```

Connect with a simple Python app

A simple python application running on the host machine can also connect to the database1.



Note: The following section assumes you already have python and redis-py (python library for connecting to Redis) configured on the host machine running the container. You can find the instructions to configure redis-py on the [github page for redis-py](#).

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

```
vi redis_test.py
```

Paste the following into a file named `redis_test.py`.

```
import redis

r = redis.StrictRedis(host='localhost', port=12000, db=0)
print ("set key1 123")
print (r.set('key1', '123'))
print ("get key1")
print(r.get('key1'))
```

Run “`redis_test.py`” application to connect to the database and store and retrieve a key using the command-line.

```
python redis_test.py
```

The output should look like the following screen if the connection is successful.

```
set key1 123
True
get key1
123
```

Test 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'))
```

Updated: August 31, 2022

Import and export data

You can [import](#), [export](#), or [back up](#) a Redis Enterprise database.

Import data

Import data from a backup or another Redis database. You can import from a single file or multiple files, such as when you want to import a backup of a clustered database.

Export data

Export data from a Redis Enterprise database to a local mount point, an FTP or SFTP server, or cloud provider storage.

Schedule automatic backups

Schedule backups of your databases to make sure you always have valid backups.

Migrate to Active-Active

Migrate a database to an [Active-Active](#) database using [Replica Of](#).

Updated: December 8, 2022

Import data into a database

You can import, [export](#), or [backup](#) files of a specific Redis Enterprise Software database to restore data. You can either import from a single file or from multiple files, such as when you want to import from a backup of a clustered database.



Warning - Importing data erases all existing content in the database.

Import data into a database

To import data into a database:

1. Select the database from the [Databases](#) list.
2. Select the target database from the list.
3. Click to open a list of additional actions.
4. Select **Import**.
5. Select the tab that corresponds to your storage location type and enter the location details.

See [Supported storage locations](#) for more information about each storage location type.

6. Select **Import**.

Supported storage locations

Data can be imported from a local mount point, transferred to a [URI](#) using FTP/SFTP, or stored on cloud provider storage.

When importing from a local mount point or a cloud provider, import locations need to be available to the group and user running Redis Enterprise Software, `redislabs:redislabs` by default.

Redis Enterprise Software needs the ability to view objects in the storage location. Implementation details vary according to the provider and your configuration. To learn more, consult the provider's documentation.

The following sections provide general guidelines. Because provider features change frequently, use your provider's documentation for the latest info.

FTP server

Before importing data from an FTP server, make sure that:

- Your Redis Enterprise cluster can connect and authenticate to the FTP server.
- The user that you specify in the FTP server location has permission to read files from the server.

To import data from an FTP server, set **RDB file path/s** using the following syntax:

```
[protocol]://[username]:[password]@[host]:[port]/[path]/[filename].rdb
```

Where:

- *protocol*: the server's protocol, can be either `ftp` or `ftps`.
- *username*: your username, if needed.
- *password*: your password, if needed.
- *hostname*: the hostname or IP address of the server.
- *port*: the port number of the server, if needed.
- *path*: the file's location path.

- *filename*: the name of the file.

Example: `ftp://username:password@10.1.1.1/home/backups/<filename>.rdb`

Select **Add path** to add another import file path.

Local mount point

Before importing data from a local mount point, make sure that:

- The node can connect to the server hosting the mount point.
- The `redislabs:redislabs` user has permission to read files on the local mount point and on the destination server.

To import from a local mount point:

1. On each node in the cluster, create the mount point:

1. Connect to the node's terminal.
2. Mount the remote storage to a local mount point.

For example:

```
sudo mount -t nfs 192.168.10.204:/DataVolume/Public /mnt/Public
```

2. In the path for the import location, enter the mount point.

For example: `/mnt/Public/<filename>.rdb`

As of version 6.2.12, Redis Enterprise reads files directly from the mount point using a [symbolic link](#) (`symlink`) instead of copying them to a temporary directory on the node.

Select **Add path** to add another import file path.

SFTP server

Before importing data from an SFTP server, make sure that:

- Your Redis Enterprise cluster can connect and authenticate to the SFTP server.
- The user that you specify in the SFTP server location has permission to read files from the server.
- The SSH private keys are specified correctly. You can use the key generated by the cluster or specify a custom key.

To use the cluster auto generated key:

1. Go to **Cluster > Security > Certificates**.
2. Expand **Cluster SSH Public Key**.
3. Download or copy the cluster SSH public key to the appropriate location on the SFTP server.

Use the server documentation to determine the appropriate location for the SSH public key.

To import data from an SFTP server, enter the SFTP server location in the format:

```
[protocol]://[username]:[password]@[host]:[port]/[path]/[filename].rdb
```

Where:

- *protocol*: the server's protocol, can be either `ftp` or `ftps`.
- *username*: your username, if needed.
- *password*: your password, if needed.
- *hostname*: the hostname or IP address of the server.
- *port*: the port number of the server, if needed.
- *path*: the file's location path.
- *filename*: the name of the file.

Example: `sftp://username:password@10.1.1.1/home/backups/[filename].rdb`

Select **Add path** to add another import file path.

AWS Simple Storage Service

Before you choose to import data from an [Amazon Web Services \(AWS\) Simple Storage Service \(S3\) bucket](#), make sure you have:

- The path to the file in your bucket in the format: s3://[bucketname]/[path]/[filename].rdb
- [Access key ID and Secret access key](#) for an [IAM user](#) with permission to read files from the bucket.

In the Redis Enterprise Software admin console, when you enter the export location details:

- Select **AWS S3**.
- In the **RDB file path/s** field, enter the path of your bucket. Select **Add path** to add another import file path.
- In the **Access key ID** field, enter the access key ID.
- In the **Secret access key** field, enter the secret access key.

You can also connect to a storage service that uses the S3 protocol but is not hosted by Amazon AWS. The storage service must have a valid SSL certificate. To connect to an S3-compatible storage location, run: `rladmin cluster config s3_url [url]`

Google Cloud Storage

Before you choose to import from a [Google Cloud](#) storage bucket, make sure that you have:

- Storage location path in the format: /bucket_name/[path]/[filename].rdb
- A [JSON service account key](#) for your account
- A [principal](#) for your bucket with the `client_email` from the service account key and the permission to get files from the bucket

In the Redis Enterprise Software admin console, when you enter the import location details:

- Select **Google Cloud Storage**.
- In the **RDB file path/s** field, enter the path of your file. Select **Add path** to add another import file path.
- In the **Client ID** field, enter the `client_id` from the service account key.
- In the **Client email** field, enter the `client_email` from the service account key.
- In the **Private key id** field, enter the `private_key_id` from the service account key.
- In the **Private key** field, enter the `private_key` from the service account key. Replace \n with new lines.

Azure Blob Storage

Before you choose to import from Azure Blob Storage, make sure that you have:

- Storage location path in the format: /container_name/[path/]/*<filename>.rdb
- Account name
- An authentication token, either an account key or an Azure [shared access signature \(SAS\)](#).

To find the account name and account key, see [Manage storage account access keys](#).

Azure SAS support requires Redis Software version 6.0.20. To learn more about Azure SAS, see [Grant limited access to Azure Storage resources using shared access signatures](#).

In the Redis Enterprise Software admin console, when you enter the import location details:

- Select **Azure Blob Storage**.
- In the **RDB file path/s** field, enter the path of your file. Select **Add path** to add another import file path.
- In the **Azure Account Name** field, enter your storage account name.
- In the **Azure Account Key** field, enter the storage account key.

Importing into an Active-Active database

When importing data into an Active-Active database, there are two options:

- [Flush all data](#) from the Active-Active database, then import the data into the database.

- Import data but merge it into the existing database.

Because Active-Active databases have a numeric counter data type, when you merge the imported data into the existing data RS increments counters by the value that is in the imported data. The import through the Redis Enterprise admin console handles these data types for you.

You can import data into an Active-Active database [from the admin console](#). When you import data into an Active-Active database, there is a special prompt warning that the imported data will be merged into the existing database.

Updated: August 17, 2023

Export data from a database

You can export the data from a specific database at any time. The following destinations are supported:

- FTP server
- SFTP server
- Amazon AWS S3
- Local mount point
- Azure Blob Storage
- Google Cloud Storage

If you export a database configured for database clustering, export files are created for each shard.

Storage space requirements

Before exporting data, verify that you have enough space available in the storage destination and on the local storage associated with the node hosting the database.

Export is a two-step process: a temporary copy of the data is saved to the local storage of the node and then copied to the storage destination. (The temporary file is removed after the copy operation.)

Export fails when there isn't enough space for either step.

Export database data

To export data from a database:

1. Select the database from the **Databases** list.
2. Click  to open a list of additional actions.
3. Select **Export**.
4. Select the tab that corresponds to your storage location type and enter the location details.

See [Supported storage locations](#) for more information about each storage location type.

5. Select **Export**.

Supported storage locations

Data can be exported to a local mount point, transferred to a [URI](#) using FTP/SFTP, or stored on cloud provider storage.

When saved to a local mount point or a cloud provider, export locations need to be available to [the group and user](#) running Redis Enterprise Software, `redislabs:redislabs` by default.

Redis Enterprise Software needs the ability to view permissions and update objects in the storage location. Implementation details vary according to the provider and your configuration. To learn more, consult the provider's documentation.

The following sections provide general guidelines. Because provider features change frequently, use your provider's documentation for the latest info.

FTP server

Before exporting data to an FTP server, verify that:

- Your Redis Enterprise cluster can connect and authenticate to the FTP server.
- The user specified in the FTP server location has permission to read and write files to the server.

To export data to an FTP server, set **Path** using the following syntax:

```
[protocol]://[username]:[password]@[host]:[port]/[path]/
```

Where:

- *protocol*: the server's protocol, can be either `ftp` or `ftps`.
- *username*: your username, if needed.
- *password*: your password, if needed.
- *hostname*: the hostname or IP address of the server.
- *port*: the port number of the server, if needed.
- *path*: the export destination path, if needed.

Example: `ftp://username:password@10.1.1.1/home/exports/`

Local mount point

Before exporting data to a local mount point, verify that:

- The node can connect to the server hosting the mount point.
- The `redislabs` user has permission to read and write files to the local mount point and to the destination server.
- The export location has enough disk space for your exported data.

To export to a local mount point:

1. On each node in the cluster, create the mount point:

1. Connect to the node's terminal.
2. Mount the remote storage to a local mount point.

For example:

```
sudo mount -t nfs 192.168.10.204:/DataVolume/Public /mnt/Public
```

2. In the path for the export location, enter the mount point.

For example: `/mnt/Public`

SFTP server

Before exporting data to an SFTP server, make sure that:

- Your Redis Enterprise cluster can connect and authenticate to the SFTP server.
- The user specified in the SFTP server location has permission to read and write files to the server.
- The SSH private keys are specified correctly. You can use the key generated by the cluster or specify a custom key.

To use the cluster auto generated key:

1. Go to **Cluster > Security > Certificates**.
2. Expand **Cluster SSH Public Key**.
3. Download or copy the cluster SSH public key to the appropriate location on the SFTP server.

Use the server documentation to determine the appropriate location for the SSH public key.

To export data to an SFTP server, enter the SFTP server location in the format:

```
sftp://[username]:[password]@[host]:[port]/[path]/
```

Where:

- *username*: your username, if needed.
- *password*: your password, if needed.
- *hostname*: the hostname or IP address of the server.
- *port*: the port number of the server, if needed.
- *path*: the export destination path, if needed.

For example: `sftp://username:password@10.1.1.1/home/exports/`

AWS Simple Storage Service

To export data to an [Amazon Web Services](#) (AWS) Simple Storage Service (S3) bucket:

1. Sign in to the [AWS console](#).
2. [Create an S3 bucket](#) if you do not already have one.
3. [Create an IAM User](#) with permission to add objects to the bucket.
4. [Create an access key](#) for that user if you do not already have one.
5. In the Redis Enterprise Software admin console, when you enter the export location details:
 - Select **AWS S3**.
 - In the **Path** field, enter the path of your bucket.
 - In the **Access key ID** field, enter the access key ID.
 - In the **Secret access key** field, enter the secret access key.

Google Cloud Storage

To export to a [Google Cloud](#) storage bucket:

1. Sign in to the Google Cloud console.
2. [Create a JSON service account key](#) if you do not already have one.
3. [Create a bucket](#) if you do not already have one.
4. [Add a principal](#) to your bucket:
 - In the **New principals** field, add the `client_email` from the service account key.
 - Select "Storage Legacy Bucket Writer" from the **Role** list.
5. In the Redis Enterprise Software admin console, when you enter the export location details:
 - Select **Google Cloud Storage**.
 - In the **Path** field, enter the path of your bucket.
 - In the **Client ID** field, enter the `client_id` from the service account key.
 - In the **Client Email** field, enter the `client_email` from the service account key.
 - In the **Private Key ID** field, enter the `private_key_id` from the service account key.
 - In the **Private key** field, enter the `private_key` from the service account key. Replace \n with new lines.

Azure Blob Storage

To export to Microsoft Azure Blob Storage, sign in to the Azure portal and then:

1. [Create an Azure Storage account](#) if you do not already have one.
2. [Create a container](#) if you do not already have one.
3. [Manage storage account access keys](#) to find the storage account name and account keys.
4. In the Redis Enterprise Software admin console, when you enter the export location details:
 - Select **Azure Blob Storage**.
 - In the **Path** field, enter the path of your bucket.
 - In the **Account name** field, enter your storage account name.
 - In the **Account key** field, enter the storage account key.

To learn more, see [Authorizing access to data in Azure Storage](#).

Flush database data

To delete the data in a database without deleting the database configuration, you can flush the data from the database.

You can use the admin console to flush data from Active-Active databases.



Warning - The flush command deletes ALL in-memory and persistence data in the database. We recommend that you [back up your database](#) before you flush the data.

Flush data from a database

From the command line, you can flush a database with the redis-cli command or with your favorite Redis client.

To flush data from a database with the redis-cli, run:

```
redis-cli -h <hostname> -p <portnumber> -a <password> flushall
```

Example:

```
redis-cli -h redis-12345.cluster.local -p 12345 -a xyz flushall
```

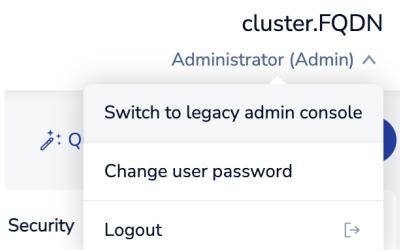
Flush data from an Active-Active database

When you flush an Active-Active database (formerly known as CRDB), all of the replicas flush their data at the same time.

To flush data from an Active-Active database:

- admin console

1. If you are using the new Cluster Manager UI, switch to the legacy admin console.



2. Go to **database** and select the Active-Active database that you want to flush.
3. Go to **configuration** and click **Flush** at the bottom of the page.
4. Enter the name of the Active-Active database to confirm that you want to flush the data.

- Command line

1. To find the ID of the Active-Active database, run:

```
crdb-cli crdb list
```

For example:

```
$ crdb-cli crdb list
CRDB-GUID          NAME      REPL-ID CLUSTER-FQDN
a16fe643-4a7b-4380-a5b2-96109d2e8bca crdb1    1      cluster1.local
a16fe643-4a7b-4380-a5b2-96109d2e8bca crdb1    2      cluster2.local
a16fe643-4a7b-4380-a5b2-96109d2e8bca crdb1    3      cluster3.local
```

2. To flush the Active-Active database, run:

```
crdb-cli crdb flush --crdb-guid <CRDB-GUID>
```

The command output contains the task ID of the flush task, for example:

```
$ crdb-cli crdb flush --crdb-guid a16fe643-4a7b-4380-a5b2-96109d2e8bca
Task 63239280-d060-4639-9bba-fc6a242c19fc created
---> Status changed: queued -> started
```

3. To check the status of the flush task, run:

```
crdb-cli task status --task-id <Task-ID>
```

For example:

```
$ crdb-cli task status --task-id 63239280-d060-4639-9bba-fc6a242c19fc
Task-ID: 63239280-d060-4639-9bba-fc6a242c19fc
CRDB-GUID: -
Status: finished
```

- REST API

1. To find the ID of the Active-Active database, use [GET /v1/crdbs](#):

```
GET https://[host][:port]/v1/crdbs
```

2. To flush the Active-Active database, use [PUT /v1/crdbs/{guid}/flush](#):

```
PUT https://[host][:port]/v1/crdbs/<guid>/flush
```

The command output contains the task ID of the flush task.

3. To check the status of the flush task, use [GET /v1/crdb_tasks](#):

```
GET https://[host][:port]/v1/crdb_tasks/<task-id>
```

Updated: August 17, 2023

Schedule periodic backups

Periodic backups provide a way to restore data with minimal data loss. With Redis Enterprise Software, you can schedule periodic backups to occur once a day (every 24 hours), twice a day (every twelve hours), every four hours, or every hour.

As of v6.2.8, you can specify the start time for 24-hour or 12-hour backups.

To make an on-demand backup, [export your data](#).

You can schedule backups to a variety of locations, including:

- FTP server
- SFTP server
- Local mount point
- Amazon Simple Storage Service (S3)
- Azure Blob Storage
- Google Cloud Storage

The backup process creates compressed (.gz) RDB files that you can [import into a database](#).

When you back up a database configured for database clustering, Redis Enterprise Software creates a backup file for each shard in the configuration. All backup files are copied to the storage location.



Note:

- Make sure that you have enough space available in your storage location. If there is not enough space in the backup location, the backup fails.
- The backup configuration only applies to the database it is configured on.
- To limit the parallel backup for shards, set both `tune cluster max_simultaneous_backups` and `tune node max_redis_forks.max_simultaneous_backups` is set to 4 by default.

Schedule periodic backups

Before scheduling periodic backups, verify that your storage location exists and is available to the user running Redis Enterprise Software (`redislabs` by default). You should verify that:

- Permissions are set correctly.
- The user running Redis Enterprise Software is authorized to access the storage location.
- The authorization credentials work.

Storage location access is verified before periodic backups are scheduled.

To schedule periodic backups for a database:

1. Sign in to the Redis Enterprise Software admin console using admin credentials.
2. From the **Databases** list, select the database, then select **Configuration**.
3. Select the **Edit** button.
4. Expand the **Scheduled backup** section.
5. Select **Add backup path** to open the **Path configuration** dialog.
6. Select the tab that corresponds to your storage location type, enter the location details, and select **Done**.

See [Supported storage locations](#) for more information about each storage location type.

7. Set the backup **Interval** and **Starting time**.

Setting	Description
Interval	Specifies the frequency of the backup; that is, the time between each backup snapshot.
Starting time	Supported values include <i>Every 24 hours</i> , <i>Every 12 hours</i> , <i>Every 4 hours</i> , and <i>Every hour</i> . <i>v6.2.8 or later</i> : Specifies the start time for the backup; available when Interval is set to <i>Every 24 hours</i> or <i>Every 12 hours</i> .
	If not specified, defaults to a time selected by Redis Enterprise Software.

8. Select **Save**.

Access to the storage location is verified when you apply your updates. This means the location, credentials, and other details must exist and function before you can enable periodic backups.

Default backup start time

If you do not specify a start time for twenty-four or twelve hour backups, Redis Enterprise Software chooses a random starting time for you.

This choice assumes that your database is deployed to a multi-tenant cluster containing multiple databases. This means that default start times are staggered (offset) to ensure availability. This is done by calculating a random offset which specifies a number of seconds added to the start time.

Here's how it works:

- Assume you're enabling the backup at 4:00 pm (1600 hours).
- You choose to back up your database every 12 hours.
- Because you didn't set a start time, the cluster randomly chooses an offset of 4,320 seconds (or 72 minutes).

This means your first periodic backup occurs 72 minutes after the time you enabled periodic backups (4:00 pm + 72 minutes). Backups repeat every twelve

hours at roughly same time.

The backup time is imprecise because they're started by a trigger process that runs every five minutes. When the process wakes, it compares the current time to the scheduled backup time. If that time has passed, it triggers a backup.

If the previous backup fails, the trigger process retries the backup until it succeeds.

In addition, throttling and resource limits also affect backup times.

For help with specific backup issues, [contact support](#).

Supported storage locations

Database backups can be saved to a local mount point, transferred to a [URI](#) using FTP/SFTP, or stored on cloud provider storage.

When saved to a local mount point or a cloud provider, backup locations need to be available to [the group and user](#) running Redis Enterprise Software, `redislabs:redislabs` by default.

Redis Enterprise Software needs the ability to view permissions and update objects in the storage location. Implementation details vary according to the provider and your configuration. To learn more, consult the provider's documentation.

The following sections provide general guidelines. Because provider features change frequently, use your provider's documentation for the latest info.

FTP server

Before enabling backups to an FTP server, verify that:

- Your Redis Enterprise cluster can connect and authenticate to the FTP server.
- The user specified in the FTP server location has read and write privileges.

To store your backups on an FTP server, set its **Backup Path** using the following syntax:

```
ftp://[username]:[password]@[host]:[port]/[path]/
```

Where:

- *protocol*: the server's protocol, can be either `ftp` or `ftps`.
- *username*: your username, if needed.
- *password*: your password, if needed.
- *hostname*: the hostname or IP address of the server.
- *port*: the port number of the server, if needed.
- *path*: the backup path, if needed.

Example: `ftp://username:password@10.1.1.1/home/backups/`

The user account needs permission to write files to the server.

SFTP server

Before enabling backups to an SFTP server, make sure that:

- Your Redis Enterprise cluster can connect and authenticate to the SFTP server.
- The user specified in the SFTP server location has read and write privileges.
- The SSH private keys are specified correctly. You can use the key generated by the cluster or specify a custom key.

To use the cluster auto generated key:

1. Go to **Cluster > Security > Certificates**.
2. Expand **Cluster SSH Public Key**.
3. Download or copy the cluster SSH public key to the appropriate location on the SFTP server.

Use the server documentation to determine the appropriate location for the SSH public key.

To backup to an SFTP server, enter the SFTP server location in the format:

```
sftp://user:password@host<:custom_port>/path/
```

For example: `sftp://username:password@10.1.1.1/home/backups/`

Local mount point

Before enabling periodic backups to a local mount point, verify that:

- The node can connect to the destination server, the one hosting the mount point.
- The `redislabs:redislabs` user has read and write privileges on the local mount point and on the destination server.
- The backup location has enough disk space for your backup files. Backup files are saved with filenames that include the timestamp, which means that earlier backups are not overwritten.

To back up to a local mount point:

1. On each node in the cluster, create the mount point:

1. Connect to a shell running on Redis Enterprise Software server hosting the node.
2. Mount the remote storage to a local mount point.

For example:

```
sudo mount -t nfs 192.168.10.204:/DataVolume/Public /mnt/Public
```

2. In the path for the backup location, enter the mount point.

For example: `/mnt/Public`

3. Verify that the user running Redis Enterprise Software has permissions to access and update files in the mount location.

AWS Simple Storage Service

To store backups in an Amazon Web Services (AWS) Simple Storage Service (S3)[bucket](#):

1. Sign in to the [AWS Management Console](#).
2. [Create an S3 bucket](#) if you do not already have one.
3. [Create an IAM User](#) with permission to add objects to the bucket.
4. [Create an access key](#) for that user if you do not already have one.
5. In the Redis Enterprise Software admin console, when you enter the backup location details:
 - Select the **AWS S3** tab on the **Path configuration** dialog.
 - In the **Path** field, enter the path of your bucket.
 - In the **Access Key ID** field, enter the access key ID.
 - In the **Secret Access Key** field, enter the secret access key.

Google Cloud Storage

For [Google Cloud](#) subscriptions, store your backups in a Google Cloud Storage bucket:

1. Sign in to the Google Cloud Platform console.
2. [Create a JSON service account key](#) if you do not already have one.
3. [Create a bucket](#) if you do not already have one.
4. [Add a principal](#) to your bucket:
 - In the **New principals** field, add the `client_email` from the service account key.
 - Select “Storage Legacy Bucket Writer” from the **Role** list.
5. In the Redis Enterprise Software admin console, when you enter the backup location details:
 - Select the **Google Cloud Storage** tab on the **Path configuration** dialog.
 - In the **Path** field, enter the path of your bucket.
 - In the **Client ID** field, enter the `client_id` from the service account key.
 - In the **Client Email** field, enter the `client_email` from the service account key.

- In the **Private Key ID** field, enter the `private_key_id` from the service account key.
- In the **Private Key** field, enter the `private_key` from the service account key. Replace \n with new lines.

Azure Blob Storage

To store your backup in Microsoft Azure Blob Storage, sign in to the Azure portal and then:

To export to Microsoft Azure Blob Storage, sign in to the Azure portal and then:

1. [Create an Azure Storage account](#) if you do not already have one.
2. [Create a container](#) if you do not already have one.
3. [Manage storage account access keys](#) to find the storage account name and account keys.
4. In the Redis Enterprise Software admin console, when you enter the backup location details:
 - Select the **Azure Blob Storage** tab on the **Path configuration** dialog.
 - In the **Path** field, enter the path of your bucket.
 - In the **Azure Account Name** field, enter your storage account name.
 - In the **Azure Account Key** field, enter the storage account key.

To learn more, see [Authorizing access to data in Azure Storage](#).

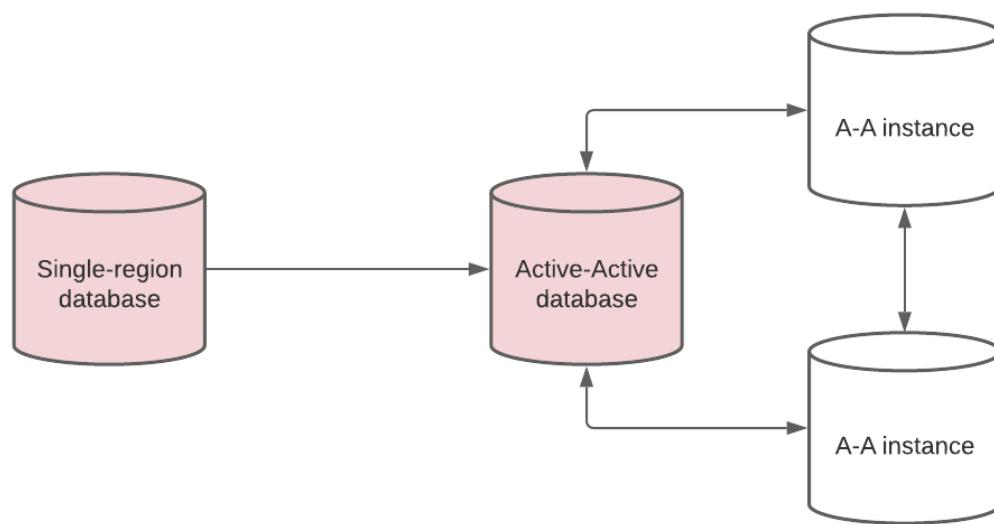
Updated: August 17, 2023

Migrate a database to Active-Active

With [Active-Active databases](#), applications can read and write to the same dataset from different geographical locations seamlessly and with latency less than 1 ms, without changing the way the application connects to the database. Active-Active databases also provide disaster recovery and accelerated data read-access for geographically distributed users.

If you have data in a single-region Redis Enterprise Software database that you want to migrate to an Active-Active database, you'll need to create a new Active-Active database and migrate the data into the new database as a [Replica Of](#) the existing database. This process will gradually populate the data in the Active-Active database.

Before data migration starts, all data is flushed from the Active-Active database. The data is migrated to the Active-Active instance where you enabled [Replica Of](#), and the data from that instance is copied to the other Active-Active instances. When data migration is finished, disable [Replica Of](#) and connect your applications to the Active-Active database.



Note: During the migration, make sure that any applications that connect to the Active-Active database are **read-only** to make sure the dataset is identical to the source database during the migration process. You may continue to write to the source database during the migration process.

To migrate a Redis Enterprise Software database to Active-Active:

1. If you are using the new Cluster Manager UI, switch to the legacy admin console.



2. Create a new [Active-Active database](#).

After the Active-Active database is activated, you see the database's configuration.

3. Click **Edit** at the bottom of the database configuration.

4. Enable **Migration using Replica Of**.

5. Click **Continue** to confirm that you want to flush the data from the Active-Active database.

6. Enter the URL of the source database endpoint (the order has no impact on replication).

- For a source database in the same RS cluster - When you click on the box, the available databases are shown in the correct format for the URL of the source endpoint:

```
<database_name>: redis://admin:<database_password>@<database_endpoint>:<database_port>
```

You can select the database that you want to use as the source.

- For a source database in a different RS cluster:

1. Log in to the Web UI of the cluster that hosts the source database.
2. In **Databases**, click on the database and go to **Configuration**.
3. Under **Endpoint**, click on **Get Replica Of source URL**.

get replica of source url

The Replica of source URL is used to configure another database as a replica of this database. [Read more](#)

```
redis://admin:3P4wUTpLUERZ8GWIZbxD736VVdyOmCh77QqQ7RjE0C1gnFW3@redis-12518.test.redislabs.com:12518
```

Cancel

Copy To Clipboard

Regenerate Password

4. Click **Copy to Clipboard** to copy the URL of the source endpoint.

If you want a different internal password, you can click **Regenerate Password**.



Warning - If you regenerate the password, replication to existing destinations fails until you update their configuration with the new password.

5. In the destination database, paste the URL of the source endpoint in the **Replica Of** box, and click .



Note: For a source database on a different Redis Enterprise Software cluster, you can [compress the replication data](#) to save bandwidth.

- For a source database in an OSS Redis cluster - Enter the URL of the source endpoint in the format:

- If the database has a password -

```
redis://:<redis_password>@<hostname>:<database_port>
```

Where the password is the Redis password represented with URL encoding escape characters.

- If the database has no password -

```
redis://<hostname>:<database_port>
```

 **Note:** If you used the mDNS protocol for the cluster name (FQDN), the [client mDNS prerequisites](#) must be met in order to communicate with other clusters.

7. Click **Update** at the bottom of the page.

8. When the synchronization icon turns green , the migration is complete. Note that migration can take minutes to hours to complete depending on the dataset size and network quality.

9. Edit the configuration of the Active-Active database and select the **Stop** button to disable **Migration using Replica Of**.



10. Redirect your database connections to the Active-Active database.

Updated: August 17, 2023

Replica Of geo-distributed Redis

In Redis Enterprise, the Replica Of feature provides active-passive geo-distribution to applications for read-only access to replicas of the data set from different geographical locations. The Redis Enterprise implementation of active-passive replication is called Replica Of.

In Replica Of, an administrator designates a database as a replica (destination) of one or more databases (sources). After the initial data load from source to destination is completed, all write commands are synchronized from the sources to the destination. Replica Of lets you distribute the read load of your application across multiple databases or synchronize the database, either within Redis Enterprise or external to Redis Enterprise, to another database.

You can [create Active-Passive](#) databases on Redis Enterprise Software or Redis Cloud.

[Active-Active Geo-Distribution \(CRDB\)](#) provides these benefits and also provides write access to all of the database replicas.

 **Warning** - Configuring a database as a replica of the database that it replicates creates a cyclical replication and is not supported.

The Replica Of is defined in the context of the destination database by specifying the source databases.

A destination database can have a maximum of thirty-two (32) source databases.

If only one source is defined, then the command execution order in the source is kept in the destination. However, when multiple sources are defined, commands that are replicated from the source databases are executed in the order in which they reach the destination database. As a result, commands that were executed in a certain order when compared across source databases might be executed in a different order on the destination database.

 **Note:** The Replica Of feature should not be confused with the in-memory [Database replication](#) feature, which is used for creating a master / replica configuration that enables ensuring database high-availability.

Replication process

When a database is defined as a replica of another database, all its existing data is deleted and replaced by data that is loaded from the source database.

Once the initial data load is completed, an ongoing synchronization process takes place to keep the destination always synchronized with its source. During the ongoing synchronization process, there is a certain delay between the time when a command was executed on the source and when it is executed on the destination. This delay is referred to as the **Lag**.

When there is a **synchronization error**, the process might stop or it might continue running on the assumption that the error automatically resolves. The result depends on the error type. See more details below.

In addition, the user can manually stop the synchronization process.

When the process is in the stopped state - whether stopped by the user or by the system - the user can restart the process. **Restarting the process causes the synchronization process to flush the DB and restart the process from the beginning.**

Replica Of status

The replication process can have the following statuses:

- **Syncing** - indicates that the synchronization process has started from scratch. Progress is indicated in percentages (%).
- **Synced** - indicates that the initial synchronization process was completed and the destination is synchronizing changes on an ongoing basis. The **Lag** delay in synchronization with the source is indicated as a time duration.
- **Sync stopped** - indicates that the synchronization process is currently not running and the user needs to restart it in order for it to continue running. This status happens if the user stops the process, or if certain errors arose that prevent synchronization from continuing without manual intervention. See more details below.

The statuses above are shown for the source database. In addition, a timestamp is shown on the source indicating when the last command from the source was executed on the destination.

The system also displays the destination database status as an aggregate of the statuses of all the sources.

 | Note: If you encounter issues with the Replica Of process, refer to the troubleshooting section [Replica Of repeatedly fails](#).

Synchronization errors

Certain errors that occur during the synchronization process require user intervention for their resolution. When such errors occur, the synchronization process is automatically stopped.

For other errors, the synchronization process continues running on the assumption that the error automatically resolves.

Examples of errors that require user intervention for their resolution and that stop the synchronization process include:

- Error authenticating with the source database.
- Cross slot violation error while executing a command on a sharded destination database.
- Out-of-memory error on a source or on the destination database.

Example of an error that does not cause the synchronization process to stop:

- Connection error with the source database. A connection error might occur occasionally, for example as result of temporary network issues that get resolved. Depending on the connection error and its duration the process might be able to start syncing from the last point it reached (partial sync) or require a complete resynchronization from scratch across all sources (full sync).

Encryption

Replica Of supports the ability to encrypt uni-directional replication communications between source and destination clusters utilizing TLS 1.2 based encryption.

Data compression for Replica Of

When the Replica Of is defined across different Redis Enterprise Software clusters, it may be beneficial to compress the data that flows through the network (depending on where the clusters physically reside and the available network).

Compressing the data reduces the traffic and can help:

- Resolve throughput issues
- Reduce network traffic costs

Compressing the data does have trade-offs, which is why it should not always be turned on by default. For example:

- It uses CPU and disk resources to compress the data before sending it to the network and decompress it on the other side.
- It takes time to compress and decompress the data which can increase latency.
- Replication is disk-based and done gradually, shard by shard in the case of a multi-shard database. This may have an impact on replication times depending on the speed of the disks and load on the database.
- If traffic is too fast and the compression takes too much time it can cause the replication process to fail and be restarted.

It is advised that you test compression out in a lower environment before enabling it in production.

In the Redis Enterprise Software management UI, when designating a Replica Of source from a different Redis Enterprise Software cluster, there is also an option to enable compression. When enabled, gzip compression with level -6 is utilized.

Database clustering (sharding) implications

If a source database is sharded, that entire database is treated as a single source for the destination database.

If the destination database is sharded, when the commands replicated from the source are executed on the destination database, the destination database's hashing function is executed to determine to which shard/s the command refers.

The source and destination can have different shard counts and functions for placement of keys.

Synchronization in Active-Passive Replication

In Active-Passive databases, one cluster hosts the source database that receives read-write operations and the other clusters host destination databases that receive synchronization updates from the source database.

When there is a significant difference between the source and destination databases, the destination database flushes all of the data from its memory and starts synchronizing the data again. This process is called a **full sync**.

For example, if the database updates for the destination databases that are stored by the destination database in a synchronization backlog exceed their allocated memory, the source database starts a full sync.

 **Warning** - When you failover to the destination database for write operations, make sure that you disable **Replica Of** before you direct clients to the destination database. This avoids a full sync that can overwrite your data.

Active-Passive replication backlog

In addition to the [database replication backlog](#), active-passive databases maintain a replication backlog (per shard) to synchronize the database instances between clusters. By default, the replication backlog is set to one percent (1%) of the database size divided by the database number of shards and ranges between 1MB to 250MB per shard. Use the [rladmin](#) utility to control the size of the replication backlog. You can set it to `auto` or set a specific size.

For an Active-Passive database:

```
rladmin tune db <db:id | name> repl_backlog <Backlog size in MB or 'auto'>
```

 **Note:** On an Active-Passive database, the replication backlog configuration applies to both the replication backlog for shards synchronization and for synchronization of database instances between clusters.

Updated: August 17, 2023

Create a database with Replica Of

Replica databases copy data from source databases (previously known as *master*), which enable read-only connections from apps and clients located in different geographic locations.

To create a replica connection, you define a database as a replica of a source database. Replica Of databases (also known as *Active-Passive databases*) synchronize in the background.

Sources databases can be:

- Located in the same Redis Enterprise Software cluster
- Located in a different Redis Enterprise cluster
- Hosted by a different deployment, e.g. Redis Enterprise Cloud
- Open source Redis (OSS) databases

Your apps can connect to the source database to read and write data; they can also use any replica for read-only access.

Replica Of can model a variety of data relationships, including:

- One-to-many relationships, where multiple replicas copy a single source database.
- Many-to-one relationships, where a single replica collects data from multiple source databases.

When you change the replica status of a database by adding, removing, or changing sources, the replica database is synchronized to the new sources.

Configure Replica Of

You can configure a database as a Replica Of, where the source database is in one of the following clusters:

- Same Redis Enterprise cluster
- Different Redis Enterprise cluster
- Open source Redis cluster

The order of the multiple Replica Of sources has no material impact on replication.

For best results when using the [Multicast DNS](#) (mDNS) protocol to resolve the fully-qualified domain name (FQDN) of the cluster, verify that your client connections meet the [client mDNS prerequisites](#).

Same Redis Enterprise cluster

To configure a Replica Of database in the same Redis Enterprise cluster as the source database:

1. [Create a new database](#) or select an existing database from the **Databases** screen.
2. For an existing database, select **Edit** from the **Configuration** tab.
3. Expand the **Replica Of** section.
4. Select **+ Add source database**.
5. In the **Connect a Replica Of source database** dialog, select **Current cluster**.
6. Select the source database from the list.
7. Select **Add source**.
8. Select **Save**.

Different Redis Enterprise cluster

To configure a Replica Of database in a different Redis Enterprise cluster from the source database:

1. Sign in to the admin console of the cluster hosting the source database.
 1. In **Databases**, select the source database and then select the **Configuration** tab.
 2. In the **Replica Of** section, select **Use this database as a source for another database**.
 3. Copy the Replica Of source URL.

Connection link to destination database



The Replica Of source URL is used to configure another database as a replica of this database.

[Read more](#)

```
redis://admin:<password>@redis-<port>.local.cluster:<port>
```

[Copy](#)

[Regenerate password](#)

To change the internal password, select [Regenerate password](#).

If you regenerate the password, replication to existing destinations fails until their credentials are updated with the new password.

2. Sign in to the admin console of the destination database's cluster.
3. [Create a new database](#) or select an existing database from the **Databases** screen.
4. For an existing database, select **Edit** from the **Configuration** tab.
5. Expand the **Replica Of** section.
6. Select **+ Add source database**.
7. In the **Connect a Replica Of source database** dialog, select **External**.
8. Enter the URL of the source database endpoint.
9. Select **Add source**.
0. Select **Save**.

For source databases on different clusters, you can[compress replication data](#) to save bandwidth.

Open source Redis cluster

To use a database from an open source Redis cluster as a Replica Of source:

1. [Create a new database](#) or select an existing database from the **Databases** screen.
2. For an existing database, select **Edit** from the **Configuration** tab.
3. Expand the **Replica Of** section.
4. Select **+ Add source database**.
5. In the **Connect a Replica Of source database** dialog, select **External**.
6. Enter the URL of the source endpoint in one of the following formats:

- For databases with passwords:

```
redis://:<password>@<host>:<port>
```

Where the password is the Redis password represented with URL encoding escape characters.

- For databases without passwords:

```
redis://<host>:<port>
```

7. Select **Add source**.

8. Select **Save**.

Configure TLS for Replica Of

When you enable TLS for Replica Of, the Replica Of synchronization traffic uses TLS certificates to authenticate the communication between the source and destination clusters.

To encrypt Replica Of synchronization traffic, configure encryption for the[replica database](#) (the destination) and the[source database](#).

Encrypt replica database traffic

To enable TLS for Replica Of in the destination database:

1. From the admin console of the cluster hosting the source database:
 1. Go to **Cluster > Security > Certificates**.
 2. Expand the **Replica Of** and **Active-Active authentication (Syncer certificate)** section.

Replica Of and Active-Active authentication (Syncer certificate)

^

Subject name:
[cluster.FQDN](#)

Issuer:
cluster.FQDN

Valid from:
20 Jul 2023 5:32 AM

Expiration date:
19 Jul 2024 5:32 AM

i

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

[Replace Certificate](#)

[Download](#) [Copy](#)

3. Download or copy the syncer certificate.
2. From the **Configuration** tab of the Replica Of destination database, select **Edit**.
3. Expand the **Replica Of** section.
4. Point to the source database entry and select to edit it.
5. Paste or upload the source syncer certificate, then select **Done**.
6. Select **Save**.

Encrypt source database traffic

To enable TLS for Replica Of cluster connections:

1. For each cluster hosting a replica:
 1. Go to **Cluster > Security > Certificates**.
 2. Expand the **Replica Of and Active-Active authentication (Syncer certificate)** section.

Replica Of and Active-Active authentication (Syncer certificate)

^

Subject name:
[cluster.FQDN](#)

Issuer:
cluster.FQDN

Valid from:
20 Jul 2023 5:32 AM

Expiration date:
19 Jul 2024 5:32 AM

i

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

[Replace Certificate](#)

[Download](#) [Copy](#)

3. Download or copy the syncer certificate.
2. From the **Security** tab of the Replica Of source database, select **Edit**.
3. In the **Secure connections (via TLS - Transport Layer Security)** section, select **Mutual TLS authentication**.

Secure connections (via TLS - Transport Layer Security)



Off

Server authentication only

Mutual TLS authentication

Communication scope



Replica Of source database



Mutual TLS (Client authentication)

Add syncer certificate from target



+ Add certificate

Additional certificate validations

No validation



+ Add validation

4. For Communication scope, select Replica Of source database.

5. Select + Add certificate, paste or upload the syncer certificate, then select Done.

Repeat this process, adding the syncer certificate for each cluster hosting a replica of this database.

6. (Optional) If you also want to require TLS for client connections, change Communication scope to Replica Of source database and clients and add client certificates.

7. Select Save.

Updated: August 17, 2023

Replica Of Repeatedly Fails

Problem: The Replica Of process repeatedly fails and restarts

Diagnostic: A log entry in the Redis log of the source database shows repeated failures and restarts.

Cause: The Redis "client-output-buffer-limit" setting on the source database is configured to a relatively small value, which causes the connection drop.

Resolution: Reconfigure the buffer on the source database to a bigger value:

- If the source is a Redis database on a Redis Enterprise Software cluster, increase the replica buffer size of the source database with:

```
rladmin tune db < db:id | name > slave_buffer < value >
```

- If the source is a Redis database not on a Redis Enterprise Software cluster, use the [config set](#) command through [redis-cli](#) to increase the client output buffer size of the **source database** with:

```
config set client-output-buffer-limit "slave <hard_limit> <soft_limit> <soft_seconds>"
```

Additional information: [Top Redis Headaches for DevOps - Replication Buffer](#)

Updated: August 31, 2022

Recover a failed database

When a cluster fails or a database is corrupted, you must:

1. [Restore the cluster configuration](#) from the CCS files
2. Recover the databases with their previous configuration and data

To restore the data that was in the databases to databases in the new cluster you must restore the database persistence files (backup, AOF, or snapshot files) to the databases. These files are stored in the [persistence storage location](#).

The database recovery process includes:

1. If the cluster failed, [recover the cluster](#).
2. Identify recoverable databases.
3. Restore the database data.
4. Verify that the databases are active.

Prerequisites

- Before you start database recovery, make sure that the cluster that hosts the database is healthy. In the case of a cluster failure, you must [recover the cluster](#) before you recover the databases.
- We recommend that you allocate new persistent storage drives for the new cluster nodes. If you use the original storage drives, make sure that you backup all files on the old persistent storage drives to another location.

Recovering the databases

After you prepare the cluster that hosts the database, you can run the recovery process from the [rladmin](#) command-line interface (CLI).

To recover the database:

1. Mount the persistent storage drives with the recovery files to the new nodes. These drives must contain the cluster configuration backup files and database persistence files.



Note: Make sure that the user `redislabs` has permissions to access the storage location of the configuration and persistence files on each of the nodes.

If you use local persistent storage, place all of the recovery files on each of the cluster nodes.

2. To see which databases are recoverable, run:

```
rladmin recover list
```

The status for each database can be either ready for recovery or missing files. An indication of missing files in any of the databases can result from:

- The storage location is not found - Make sure that on all of the nodes in the cluster the recovery path is set correctly.
- Files are not found in the storage location - Move the files to the storage location.
- No permission to read the files - Change the file permissions so that `redislabs:redislabs` has 640 permissions.
- Files are corrupted - Locate copies of the files that are not corrupted.

If you cannot resolve the issues, contact [Redis support](#).

3. Recover the database, either:

- Recover the databases all at once from the persistence files located in the persistent storage drives: `rladmin recover all`
- Recover a single database from the persistence files located in the persistent storage drives: `rladmin recover db <database_id|name>`
- Recover only the database configuration for a single database (without the data): `recover db only_configuration <db_name>`



Note:

- If persistence was not configured for the database, the database is restored empty.
- For Active-Active databases that still have live instances, we recommend that you recover the configuration for the failed instances and let the data update from the other instances.
- For Active-Active databases that all instances need to be recovered, we recommend that you recover one instance with the data and only recover the configuration for the other instances. The empty instances then update from the recovered data.
- If the persistence files of the databases from the old cluster are not stored in the persistent storage location of the new node, you must first map the recovery path of each node to the location of the old persistence files. To do this, run the `node <id> recovery_path set` command in `rladmin`. The persistence files for each database are located in the persistent storage path of the nodes from the old cluster, usually under `/var/opt/redislabs/persist/redis`.

4. To verify that the recovered databases are now active, run: `rladmin status`

After the databases are recovered, make sure that your Redis clients can successfully connect to the databases.

Updated: August 17, 2023

Delete databases

When you delete a database, both the database configuration and data are removed.

To delete a database from the admin console:

1. From the **Databases** list, select the database, then select **Configuration**.
2. Select **Delete**.
3. In the **Delete database** dialog, confirm deletion.

Updated: August 17, 2023

Active-Active geo-distributed Redis

In Redis Enterprise, Active-Active geo-distribution is based on [CRDT technology](#). The Redis Enterprise implementation of CRDT is called an Active-Active database (formerly known as CRDB). With Active-Active databases, applications can read and write to the same data set from different geographical locations seamlessly and with latency less than one millisecond (ms), without changing the way the application connects to the database.

Active-Active databases also provide disaster recovery and accelerated data read-access for geographically distributed users.

High availability

The [high availability](#) that Active-Active replication provides is built upon a number of Redis Enterprise Software features (such as [clustering](#), [replication](#), and [replica HA](#)) as well as some features unique to Active-Active ([multi-primary replication](#), [automatic conflict resolution](#), and [strong eventual consistency](#)).

Clustering and replication are used together in Active-Active databases to distribute multiple copies of the dataset across multiple nodes and multiple clusters. As a result, a node or cluster is less likely to become a single point of failure. If a primary node or primary shard fails, a replica is automatically promoted to primary. To avoid having one node hold all copies of certain data, the [replica HA](#) feature (enabled by default) automatically migrates replica shards to available nodes.

Multi-primary replication

In Redis Enterprise Software, replication copies data from primary shards to replica shards. Active-Active geo-distributed replication also copies both primary and replica shards to other clusters. Each Active-Active database needs to span at least two clusters; these are called participating clusters.

Each participating cluster hosts an instance of your database, and each instance has its own primary node. Having multiple primary nodes means you can connect to the proxy in any of your participating clusters. Connecting to the closest cluster geographically enables near-local latency. Multi-primary

replication (previously referred to as multi-master replication) also means that your users still have access to the database if one of the participating clusters fails.



Note: Active-Active databases do not replicate the entire database, only the data. Database configurations, LUA scripts, and other support info are not replicated.

Syncer

Keeping multiple copies of the dataset consistent across multiple clusters is no small task. To achieve consistency between participating clusters, Redis Active-Active replication uses a process called the [syncer](#).

The syncer keeps a [replication backlog](#), which stores changes to the dataset that the syncer sends to other participating clusters. The syncer uses partial syncs to keep replicas up to date with changes, or a full sync in the event a replica or primary is lost.

Conflict resolution

Because you can connect to any participating cluster to perform a write operation, concurrent and conflicting writes are always possible. Conflict resolution is an important part of the Active-Active technology. Active-Active databases only use [conflict-free replicated data types \(CRDTs\)](#). These data types provide a predictable conflict resolution and don't require any additional work from the application or client side.

When developing with CRDTs for Active-Active databases, you need to consider some important differences. See [Develop applications with Active-Active databases](#) for related information.

Strong eventual consistency

Maintaining strong consistency for replicated databases comes with tradeoffs in scalability and availability. Redis Active-Active databases use a strong eventual consistency model, which means that local values may differ across replicas for short periods of time, but they all eventually converge to one consistent state. Redis uses vector clocks and the CRDT conflict resolution to strengthen consistency between replicas. You can also enable the causal consistency feature to preserve the order of operations as they are synchronized among replicas.

Other Redis Enterprise Software features can also be used to enhance the performance, scalability, or durability of your Active-Active database. These include [data persistence](#), [multiple active proxies](#), [distributed synchronization](#), [the OSS Cluster API](#), and [rack-zone awareness](#).

Next steps

- [Plan your Active-Active deployment](#)
- [Get started with Active-Active](#)
- [Create an Active-Active database](#)

Updated: August 31, 2022

Get started with Redis Enterprise Active-Active databases

To get you started, this article will help you 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 [Redis Enterprise Software download package](#), set up two Redis Enterprise Software 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](#).

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.

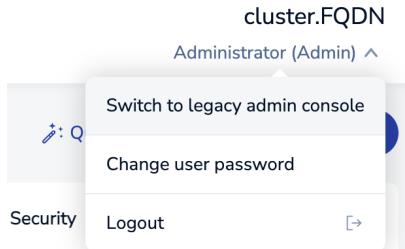
Set up two clusters

1. For cluster 1, direct your browser to <https://localhost:8443> on the host machine to see the Redis Enterprise Software web console.



Note: Depending on your browser, you may see a certificate error. Continue to the website.

If you are using the new Cluster Manager UI, switch to the legacy admin console.



2. Click the **Setup** button.

redis enterprise
by redis labs
Version 6.0.8-30

Setup

3. On the **node configuration** page, select your default settings and provide a cluster FQDN, for example `cluster1.local`. Then click **Next** button.

node configuration

Persistent storage path	/var/opt/redislabs/persist		Free = 43.02 GB
Ephemeral storage path (i)	/var/opt/redislabs/tmp		Free = 43.02 GB
<input type="checkbox"/> Enable flash storage support (i) Read more			
IP Addresses Usage (i) Read more	IP Address	Interface	Internal traffic (i)
	IPv4		
	172.17.0.2	eth0	<input checked="" type="radio"/>
			<input checked="" type="checkbox"/>
+			
Cluster configuration	<input checked="" type="radio"/> Create new cluster <input type="radio"/> Join cluster		
Cluster name (FQDN) Read more	cluster.local		
<input type="checkbox"/> Enable private & public endpoints support Read more			
<input type="checkbox"/> Enable rack-zone awareness Read more			

[Next](#)

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

5. On the next screen, set up a Cluster Administrator account using an email for the login and a password.

set admin credentials

Email	admin@redislabs.com
Password	***** (i)
Verify password	*****

[Back](#)

[Next](#)

6. Click **OK** to confirm that you are aware of the replacement of the HTTPS SSL/TLS certificate on the node, and proceed through the browser warning.

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**.

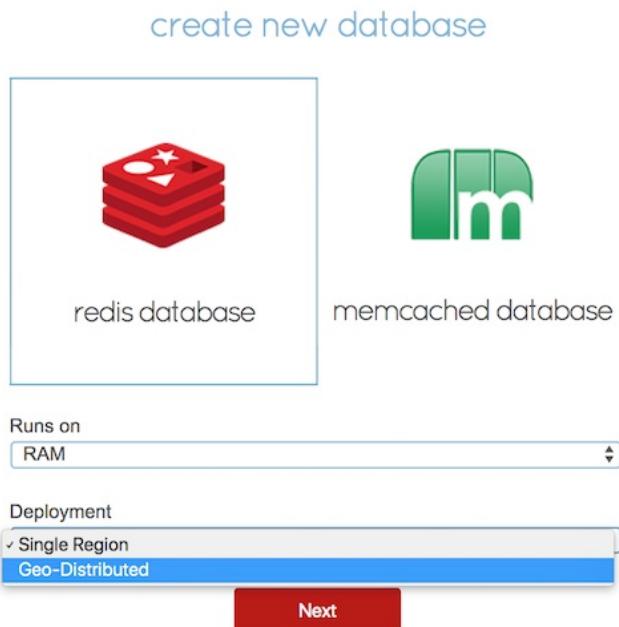
Now we have two Redis Enterprise Software clusters with FQDNs **cluster1.local** and **cluster2.local**.



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

Create an 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.

create database

Name	database1									
Protocol	Redis									
Runs on	RAM									
Memory limit (GB) <small>Read more</small>	0.1 GB 2.30 GB RAM unallocated									
<input type="checkbox"/> Replication <small>i</small>										
Redis Modules	<small>+</small>									
Data persistence	None									
<input checked="" type="checkbox"/> Default database access <small>i</small>	Password Confirm password									
Endpoint port number	12000									
<input checked="" type="checkbox"/> Database clustering <small>Read more</small>	Number of shards 1 <small>+ -</small>									
<input type="checkbox"/> OSS Cluster API support <small>Read more</small>										
Eviction policy	noeviction									
Participating Clusters <small>i</small> <small>Read more</small>	<table border="1"><tr><td>https://cluster1.local:9443</td><td>admin@redislabs.com</td><td>*****</td></tr><tr><td>https://cluster2.local:9443</td><td>admin@redislabs.com</td><td>*****</td></tr><tr><td><small>+</small></td><td></td><td></td></tr></table>	https://cluster1.local:9443	admin@redislabs.com	*****	https://cluster2.local:9443	admin@redislabs.com	*****	<small>+</small>		
https://cluster1.local:9443	admin@redislabs.com	*****								
https://cluster2.local:9443	admin@redislabs.com	*****								
<small>+</small>										
<input type="checkbox"/> Causal Consistency across replicas <small>Read more</small>	<small>OK</small> <small>✖️</small> <small>✖️</small> <small>✖️</small>									



Note: If you cannot activate the database because of a memory limitation, make sure that Docker has enough memory allocated in the Docker Settings.

4. 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>.

5. 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.

Test connection

With the Redis database created, you are ready to connect to your database. See [Connect to Active-Active databases](#) for tutorials and examples of multiple connection methods.

Updated: August 17, 2023

Considerations for planning Active-Active databases

In Redis Enterprise, Active-Active geo-distribution is based on [conflict-free replicated data type \(CRDT\) technology](#). Compared to databases without geo-distribution, Active-Active databases have more complex replication and networking, as well as a different data type.

Because of the complexities of Active-Active databases, there are special considerations to keep in mind while planning your Active-Active database.

See [Active-Active Redis](#) for more information about geo-distributed replication. For more info on other high availability features, see [Durability and high availability](#).

Participating clusters

For Active-Active databases, you need to [set up your participating clusters](#). At least two participating clusters. If your database requires more than ten participating clusters, contact Redis support. You can [add or remove participating clusters](#) after database creation.

Changes made from the admin console to an Active-Active database configuration only apply to the cluster you are editing. For global configuration changes across all clusters, use the `crdb-cli` command-line utility.

Memory limits

Database memory limits define the maximum size of your database across all database replicas and [shards](#) on the cluster. Your memory limit also determines the number of shards.

Besides your dataset, the memory limit must also account for replication, Active-Active metadata, and module overhead. These features can increase your database size, sometimes increasing it by two times or more.

Factors to consider when sizing your database:

- **dataset size:** you want your limit to be above your dataset size to leave room for overhead.
- **database throughput:** high throughput needs more shards, leading to a higher memory limit.
- **modules:** using modules with your database can consume more memory.
- **database clustering:** enables you to spread your data into shards across multiple nodes (scale out).
- **database replication:** enabling replication doubles memory consumption
- **Active-Active replication:** enabling Active-Active replication requires double the memory of regular replication, which can be up to two times (2x) the original data size per instance.
- **database replication backlog** for synchronization between shards. By default, this is set to 1% of the database size.
- **Active-Active replication backlog** for synchronization between clusters. By default, this is set to 1% of the database size.

It's also important to know Active-Active databases have a lower threshold for activating the eviction policy, because it requires propagation to all participating clusters. The eviction policy starts to evict keys when one of the Active-Active instances reaches 80% of its memory limit.

For more information on memory limits, see [Memory and performance](#) or [Database memory limits](#).

Networking

Network requirements for Active-Active databases include:

- A VPN between each network that hosts a cluster with an instance (if your database spans WAN).
- A network connection to [several ports](#) on each cluster from all nodes in all participating clusters.
- A [network time service](#) running on each node in all clusters.

Networking between the clusters must be configured before creating an Active-Active database. The setup will fail if there is no connectivity between the clusters.

Network ports

Every node must have access to the REST API ports of every other node as well as other ports for proxies, VPNs, and the admin console. See [Network port configurations](#) for more details. These ports should be allowed through firewalls that may be positioned between the clusters.

Network Time Service

Active-Active databases require a time service like NTP or Chrony to make sure the clocks on all cluster nodes are synchronized. This is critical to avoid problems with internal cluster communications that can impact your data integrity.

See [Synchronizing cluster node clocks](#) for more information.

Redis modules

Several Redis modules are compatible with Active-Active databases. Find the list of [compatible Redis modules](#).



Note: Starting with v6.2.18, you can index, query, and perform full-text searches of nested JSON documents in Active-Active databases by combining RedisJSON and RediSearch.

Limitations

Active-Active databases have the following limitations:

- An existing database can't be changed into an Active-Active database. To move data from an existing database to an Active-Active database, you must [create a new Active-Active database](#) and [migrate the data](#).
- [Discovery service](#) is not supported with Active-Active databases. Active-Active databases require FQDNs or [mDNS](#).
- The FLUSH command is not supported from the CLI. To flush your database, use the API or admin console.
- Cross slot multi commands (such as MSET) are not supported with Active-Active databases.
- The hashing policy can't be changed after database creation.

Updated: August 17, 2023

Create an Active-Active geo-replicated database

Active-Active geo-replicated databases (formerly known as CRDBs) give applications write access to replicas of the dataset in different geographical locations.

The participating Redis Enterprise Software clusters that host the instances can be in [distributed geographic locations](#). Every instance of an Active-Active database can receive write operations, and all operations are [synchronized](#) to all of the instances without conflict.

Steps to create an Active-Active database

1. [Create a service account](#) - On each participating cluster, create a dedicated user account with the Admin role.
2. [Confirm connectivity](#) - Confirm network connectivity between the participating clusters.
3. [Create Active-Active database](#) - Connect to one of your clusters and create a new Active-Active database.
4. [Add participating clusters](#) - Add the participating clusters to the Active-Active database with the user credentials for the service account.
5. [Verify creation](#) - Log in to each of the participating clusters and verify your Active-Active database was created on them.
6. [Confirm Active-Active database synchronization](#) - Test writing to one cluster and reading from a different cluster.

Prerequisites

- Two or more machines with the same version of RS installed
- Network connectivity and cluster FQDN name resolution between all participating clusters
- [Network time service](#) listener (ntpd) configured and running on each node in all clusters

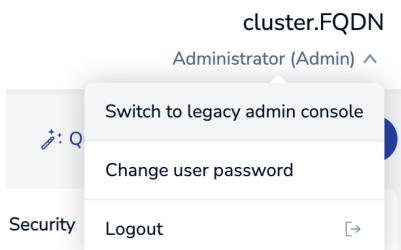
Create an Active-Active database

1. To create service accounts, on each participating cluster:

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

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

If you are using the new Cluster Manager UI, switch to the legacy admin console.



2. Go to [access control > users](#) and select .
3. Enter the name, email, and password for the user, select the [Admin](#) role, and select .

access control

users roles redis acls

User Name	Email	Role	Authentication	Password	Email Alerts
Administrator	admin@redislabs.com	Admin	internal	*****	All
ServiceAccount	service@redislabs.com	Admin	internal	new password confirm password	None Edit   

2. To make sure that there is network connectivity between the participating clusters, telnet on port 9443 from each participating cluster to each of the other participating clusters.

```
telnet <target FQDN> 9443
```

3. In your web browser, open the admin console of the cluster that you want to connect to in order to create the Active-Active database. By default, the address is: https://<RS_address>:8443

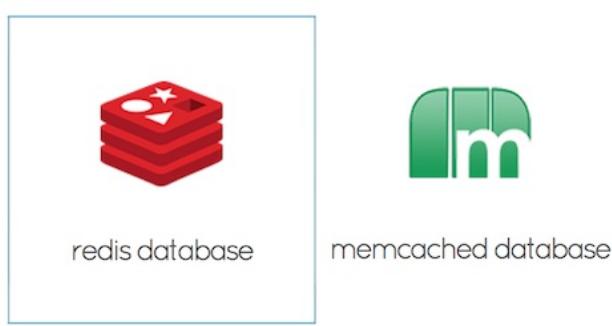
4. In **databases**, click .

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

5. In the **Deployment** box, select **Geo-Distributed** and click **Next** to create an Active-Active database on RAM.

If your cluster supports **Auto Tiering**, in **Runs on** you can select **Flash** so that your database uses Flash memory. We recommend that you use AOF every 1 sec for the best performance during the initial Active-Active database sync of a new replica.

create new database



Runs on
RAM

Deployment
 Single Region
 Geo-Distributed

Next

6. Enter the name of the new Active-Active database and select from the options:



Note:

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

- **Replication** - We recommend that all Active-Active database use replication for best intercluster synchronization performance. When replication is enabled, every Active-Active database master shard is replicated to a corresponding replica shard. The replica shards are then used to synchronize data between the instances, and the master shards are dedicated to handling client requests. We also recommend that you enable [replica HA](#) to ensure that the replica shards are highly-available for this synchronization.
- **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 (AOF). AOF provides the fastest and most reliable method for instance failure recovery.
- **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.

7. Configure the 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. You can only configure access control after the Active-Active database is created.

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 (Required)** - The port in the range 10000-19999 that clients must use to connect to the Active-Active database.

- In the **Database clustering** option, you can either:

- Make sure the 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 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.

- **Eviction policy** - The default eviction policy for Active-Active databases is [noeviction](#). Redis Enterprise version 6.0.20 and later support all eviction policies for Active-Active databases, unless [Auto Tiering](#) is enabled.
- **Participating Clusters** - You must specify the URL of the clusters that you want to host instances of an Active-Active database and the admin user account to connect to each cluster.
 1. In the **Participating Clusters** list, click  to add clusters.
 2. For each cluster, enter the URL for the cluster (https://<cluster_fqdn>:9443), enter the credentials (email address and password) for the service account that you created, and click .
- **Causal Consistency** - Causal Consistency in an Active-Active database guarantees that the order of operations on a specific key is maintained across all instances of an Active-Active database. To enable Causal Consistency for an existing Active-Active database, use the REST API.
- **TLS** - If you enable TLS when you create the Active-Active database, the nodes use the TLS mode **Require TLS for CRDB communication only** to require TLS authentication and encryption for communications between participating clusters. After you create the Active-Active database, you can set the TLS mode to **Require TLS for all communications** so that client communication from applications are also authenticated and encrypted.

Test the connection to your member Redis Active-Active databases

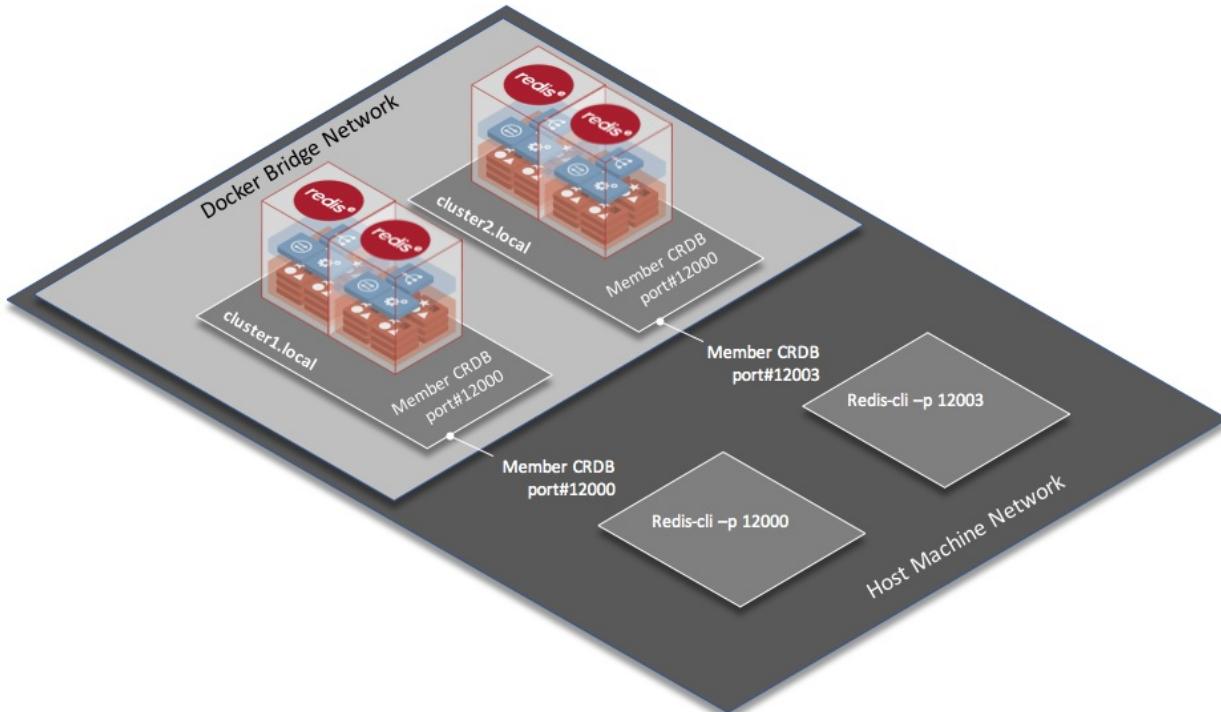
With the Redis database created, you are ready to connect to your database. See [Connect to Active-Active databases](#) for tutorials and examples of multiple connection methods.

Connect to your 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 use redis-cli on port 12000 from the node 1 terminal, run:

```
redis-cli -p 12000
```

2. 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"
```

3. Enter the terminal of node 1 in cluster 2, run the redis-cli, and retrieve key1.

The output of the commands looks like this:

```
$ 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.

```
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: August 31, 2022

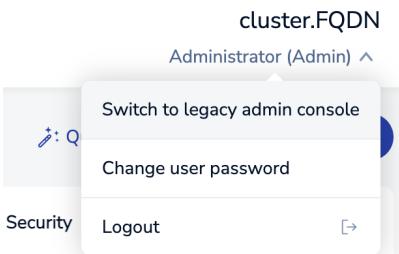
Manage Active-Active databases

You can configure and manage your Active-Active database from either the admin console or the command line.

To change the global configuration of the Active-Active database, use the [crdb-cli](#).

If you need to apply changes locally to one database instance, you use the admin console or the [rladmin](#) CLI.

If you are using the new Cluster Manager UI, switch to the legacy admin console to manage Active-Active databases.



Database settings

The following table shows a list of database settings, tools you can use to change those settings, and links to more information.

Much of the Active-Active database settings can be changed after the database has been created. One notable exception is database clustering. Database clustering can't be turned on or off after the database has been created and will remain the same through the lifetime of the database.

Participating clusters

You can add and remove participating clusters of an Active-Active database to change the topology. To manage the changes to Active-Active topology, use the [crdb-cli](#) tool or the participating clusters list in the admin console. You can make multiple changes at once and changes will be committed when the database configuration is saved.

Participating Clusters (i) Read more	https://cluster1.local:9443	admin@redislabs.com	*****
	https://cluster2.local:9443	admin@redislabs.com	*****
	https://cluster3.local:9443	admin@redislabs.com

[+](#)

Add participating clusters

All of the existing participating clusters must be online and in a syncing state when you add new participating clusters.

New participating clusters create the Active-Active database instance based on the global Active-Active database configuration. After you add new participating clusters to an existing Active-Active database, the new database instance can accept connections and read operations. The new instance does not accept write operations until it is in the syncing state.

Remove participating clusters

All of the existing participating clusters must be online and in a syncing state when you remove an online participating clusters. If you must remove offline participating clusters, you can do this with forced removal. If a participating cluster that was removed forcefully returns attempts to re-join the cluster, it will have an out of date on Active-Active database membership. The joined participating clusters reject updates sent from the removed participating cluster. To avoid re-join attempts, purge the forcefully removed instance from the participating cluster.

Replication backlog

Redis databases that use [replication for high availability](#) maintain a replication backlog (per shard) to synchronize the primary and replica shards of a database. In addition to the database replication backlog, Active-Active databases maintain a backlog (per shard) to synchronize the database instances between clusters.

By default, both the database and Active-Active replication backlogs are set to one percent (1%) of the database size divided by the number of shards. This can range between 1MB to 250MB per shard for each backlog.

Change the replication backlog size

Use the [crdb-cli](#) utility to control the size of the replication backlogs. You can set it to `auto` or set a specific size.

Update the database replication backlog configuration with the `crdb-cli` command shown below.

```
crdb-cli crdb update --crdb-guid <crdb_guid> --default-db-config "{\"repl_backlog_size\": <size in MB | 'auto'>}"
```

Update the Active-Active (CRDT) replication backlog with the command shown below:

```
crdb-cli crdb update --crdb-guid <crdb_guid> --default-db-config "{\"crdt_repl_backlog_size\": <size in MB | 'auto'>}"
```

Data persistence

Active-Active supports AOF (Append-Only File) data persistence only. Snapshot persistence is *not* supported for Active-Active databases and should not be used.

If an Active-Active database is currently using snapshot data persistence, use `crdb-cli` to switch to AOF persistence:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> --default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

Updated: August 17, 2023

Delete Active-Active databases

When you delete an Active-Active database (formerly known as CRDB), all instances of the Active-Active database are deleted from all participating clusters.

 | **Warning** - This action is immediate, non-reversible, and has no rollback.

Because Active-Active databases are made up of instances on multiple participating clusters, to restore a deleted Active-Active database you must create the database again with all of its instances and then restore the data to the database from backup.

We recommended that you:

- Back up your data and test the restore on another database before you delete an Active-Active database.
- Consider [flushing the data](#) from the database so that you can keep the Active-Active database configuration and restore the data to it if necessary.

Updated: August 31, 2022

Enable causal consistency

When you enable causal consistency in Active-Active databases, the order of operations on a specific key are maintained across all Active-Active database instances.

For example, if operations A and B were applied on the same key and the effect of A was observed by the instance that initiated B before B was applied to the key. All instances of an Active-Active databases would then observe the effect of A before observing the effect of B. This way, any causal relationship between operations on the same key is also observed and maintained by every replica.

Enable causal consistency

When you create an Active-Active database, causal consistency is set as follows in the legacy admin console:

create database

Name	online-orders						
Protocol	Redis						
Runs on	RAM						
Memory limit (GB) <small>Read more</small>	0.1 GB 3.45 GB RAM unallocated						
<input checked="" type="checkbox"/> Replication <small>i</small>							
Redis Modules	+						
Data persistence	None						
<input checked="" type="checkbox"/> Default database access <small>i</small>							
Endpoint port number	<input type="text"/>						
<input checked="" type="checkbox"/> Database clustering <small>Read more</small>	Number of shards 2 + - 4 with replication						
<input type="checkbox"/> OSS Cluster API support <small>Read more</small>							
Eviction policy	noeviction						
Participating Clusters <small>i</small> <small>Read more</small>	<table><tr><td>https://cluster1.local:9443</td><td>admin@redislabs.com</td><td>*****</td></tr><tr><td>https://cluster2.local:9443</td><td>admin@redislabs.com</td><td>*****</td></tr></table>	https://cluster1.local:9443	admin@redislabs.com	*****	https://cluster2.local:9443	admin@redislabs.com	*****
https://cluster1.local:9443	admin@redislabs.com	*****					
https://cluster2.local:9443	admin@redislabs.com	*****					
<input checked="" type="checkbox"/> Causal Consistency across replicas <small>Read more</small>	+						
<input type="checkbox"/> TLS <small>Read more</small>							

[Cancel](#)

[Activate](#)

Once enabled, additional operations to enable or disable can only be performed using the REST API or the `rdb-cli` tool. In this case, the updated Active-Active database behavior happens only for commands and operations received after the change.

Causal consistency side effects

When the causal consistency option is enabled, each instance maintains the order of operations it received from another instance and relays that information to all other N-2 instances, where N represents the number of instances used by the Active-Active database.

As a result, network traffic is increased by a factor of (N-2). The memory consumed by each instance and overall performance are also impacted when causal consistency is activated.

Updated: August 17, 2023

Configure distributed synchronization

Replicated databases, such as [Replica Of](#) and [Active-Active](#) databases, use proxy endpoints to synchronize database changes with the databases on other participating clusters.

To improve the throughput and lower the latency for synchronization traffic, you can configure a replicated database to use distributed synchronization where any available proxy endpoint can manage synchronization traffic.

Every database by default has one proxy endpoint that manages client and synchronization communication with the database shards, and that proxy endpoint is used for database synchronization. This is called centralized synchronization.

To prepare a database to use distributed synchronization you must first make sure that the database proxy policy is defined so that either each node has a proxy endpoint or each primary (master) shard has a proxy endpoint. After you have multiple proxies for the database, you can configure the database synchronization to use distributed synchronization.

Configure distributed synchronization

To configure distributed synchronization:

1. To check the proxy policy for the database, run: `rladmin status`

The output of the status command shows the list of endpoints on the cluster and the proxy policy for the endpoint.

ENDPOINTS:					
DB:ID	NAME	ID	NODE	ROLE	SSL
db:1	db	endpoint:1:1	node:1	all-master-shards	No

If the proxy policy (also known as a role) is single, configure the policy to all-nodes or all-master-shards according to your needs with the command:

```
rladmin bind db <db_name> endpoint <endpoint id> policy <all-master-shards|all-nodes>
```

2. To configure the database to use distributed synchronization, run:

```
rladmin tune db <db_name> syncer_mode distributed
```

To change back to centralized synchronization, run:

```
rladmin tune db <db_name> syncer_mode centralized
```

Verify database synchronization

Use `rladmin` to verify a database synchronization role:

```
rladmin info db <db_name>
```

The current database role is reported as the syncer_mode value:

```
$ rladmin info db <db_name>
db:1 [<db_name>]:
// (Other settings removed)
syncer_mode: centralized
```

Updated: August 17, 2023

Syncer process

Syncer process

Each node in a cluster containing an instance of an Active-Active database hosts a process called the syncer. The syncer process:

1. Connects to the proxy on another participating cluster
2. Reads data from that database instance
3. Writes the data to the local cluster's primary(master) shard

Some replication capabilities are also included in [open source redis](#).

The primary (also known as master) shard at the top of the primary-replica tree creates a replication ID. This replication ID is identical for all replicas in that tree. When a new primary is appointed, the replication ID changes, but a partial sync from the previous ID is still possible.

In a partial sync, the backlog of operations since the offset are transferred as raw operations. In a full sync, the data from the primary is transferred to the

replica as an RDB file which is followed by a partial sync.

Partial synchronization requires a backlog large enough to store the data operations until connection is restored. See [replication backlog](#) for more info on changing the replication backlog size.

Syncer in Active-Active replication

In the case of an Active-Active database:

- Multiple past replication IDs and offsets are stored to allow for multiple syncs
- The [Active-Active replication backlog](#) is also sent to the replica during a full sync.



Warning - Full sync triggers heavy data transfers between geo-replicated instances of an Active-Active database.

An Active-Active database uses partial synchronization in the following situations:

- Failover of primary shard to replica shard
- Restart or crash of replica shard that requires sync from primary
- Migrate replica shard to another node
- Migrate primary shard to another node as a replica using failover and replica migration
- Migrate primary shard and preserve roles using failover, replica migration, and second failover to return shard to primary



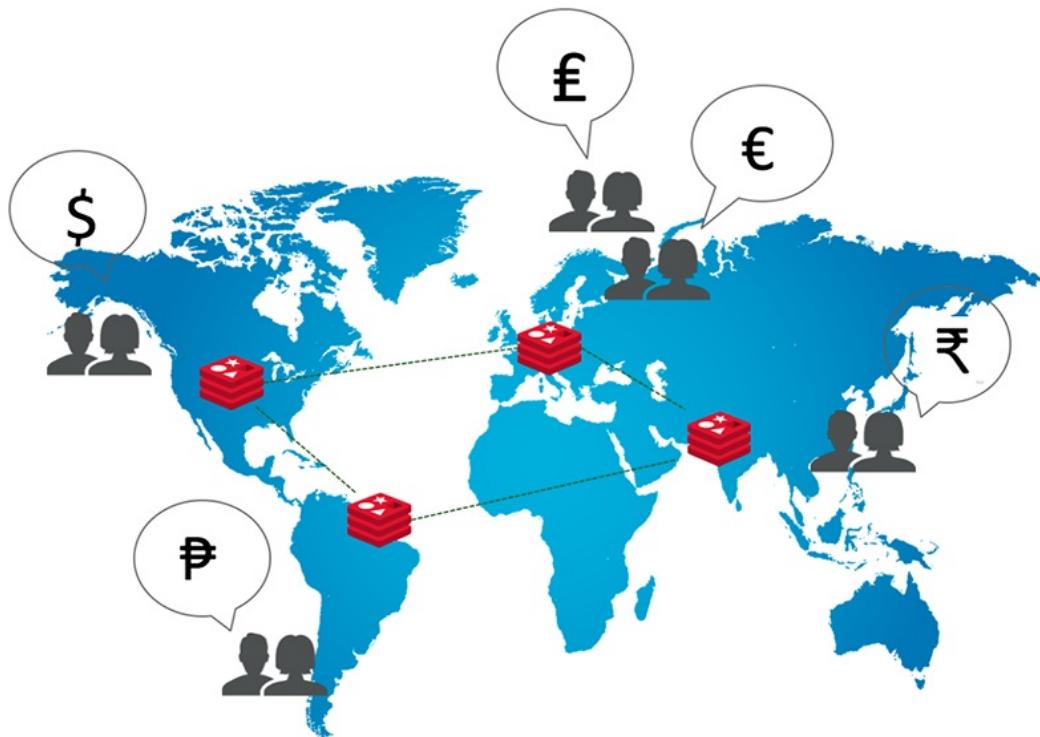
Note: Synchronization of data from the primary shard to the replica shard is always a full synchronization.

Updated: August 31, 2022

Active-Active Redis applications

Developing globally distributed applications can be challenging, as developers have to think about race conditions and complex combinations of events under geo-failovers and cross-region write conflicts. In Redis Enterprise Software (RS), Active-Active databases simplify developing such applications by directly using built-in smarts for handling conflicting writes based on the data type in use. Instead of depending on just simplistic “last-writer-wins” type conflict resolution, geo-distributed Active-Active databases (formerly known as CRDBs) combines techniques defined in CRDT (conflict-free replicated data types) research with Redis types to provide smart and automatic conflict resolution based on the data types intent.

An Active-Active database is a globally distributed database that spans multiple Redis Enterprise Software clusters. Each Active-Active database can have many Active-Active database instances that come with added smarts for handling globally distributed writes using the proven [CRDT](#) approach. [CRDT](#) research describes a set of techniques for creating systems that can handle conflicting writes. CRDBs are powered by Multi-Master Replication (MMR) provides a straightforward and effective way to replicate your data between regions and simplify development of complex applications that can maintain correctness under geo-failovers and concurrent cross-region writes to the same data.



Active-Active databases replicate data between multiple Redis Enterprise Software clusters. Common uses for Active-Active databases include disaster recovery, geographically redundant applications, and keeping data closer to your user's locations. MMR is always multi-directional amongst the clusters configured in the Active-Active database. For unidirectional replication, please see the [Replica Of](#) capabilities in Redis Enterprise Software.

Example of synchronization

In the example below, database writes are concurrent at the point in times t1 and t2 and happen before a sync can communicate the changes. However, writes at times t4 and t6 are not concurrent as a sync happened in between.

Time	CRDB Instance1	CRDB Instance2
t1	SET key1 "a"	
t2		SET key1 "b"
t3	– Sync –	– Sync –
t4	SET key1 "c"	
t5	– Sync –	– Sync –
t6		SET key1 "d"

[Learn more about synchronization](#) for each supported data type and [how to develop](#) with them on Redis Enterprise Software.

Updated: August 31, 2022

Develop applications with Active-Active databases

Developing geo-distributed, multi-master applications can be difficult. Application developers may have to understand a large number of race conditions between updates to various sites, network, and cluster failures that could reorder the events and change the outcome of the updates performed across geo-distributed writes.

Active-Active databases (formerly known as CRDB) are geo-distributed databases that span multiple Redis Enterprise Software (RS) clusters. Active-Active databases depend on multi-master replication (MMR) and Conflict-free Replicated Data Types (CRDTs) to power a simple development experience for geo-distributed applications. Active-Active databases allow developers to use existing Redis data types and commands, but understand the developer's intent and automatically handle conflicting concurrent writes to the same key across multiple geographies. For example, developers can simply use the INCR or INCRBY method in Redis in all instances of the geo-distributed application, and Active-Active databases handle the additive nature of INCR to reflect the correct final value. The following example displays a sequence of events over time : t1 to t9. This Active-Active database has two member Active-Active databases : member CRDB1 and member CRDB2. The local operations executing in each member Active-Active database is listed under the member Active-Active database name. The "Sync" even represent the moment where synchronization catches up to distribute all local member Active-Active database updates to other participating clusters and other member Active-Active databases.

Time	Member CRDB1	Member CRDB2
t1	INCRBY key1 7	
t2		INCRBY key1 3
t3	GET key1 7	GET key1 3
t4	– Sync –	– Sync –
t5	GET key1 10	GET key1 10
t6	DECRBY key1 3	
t7		INCRBY key1 6
t8	– Sync –	– Sync –
t9	GET key1 13	GET key1 13

Databases provide various approaches to address some of these concerns:

- Active-Passive Geo-distributed deployments: With active-passive distributions, all writes go to an active cluster. Redis Enterprise provides a “Replica Of” capability that provides a similar approach. This can be employed when the workload is heavily balanced towards read and few writes. However, WAN performance and availability is quite flaky and traveling large distances for writes take away from application performance and availability.
- Two-phase Commit (2PC): This approach is designed around a protocol that commits a transaction across multiple transaction managers. Two-phase commit provides a consistent transactional write across regions but fails transactions unless all participating transaction managers are “available” at the time of the transaction. The number of messages exchanged and its cross-regional availability requirement make two-phase commit unsuitable for even moderate throughputs and cross-geo writes that go over WANs.
- Sync update with Quorum-based writes: This approach synchronously coordinates a write across majority number of replicas across clusters spanning multiple regions. However, just like two-phase commit, number of messages exchanged and its cross-regional availability requirement make geo-distributed quorum writes unsuitable for moderate throughputs and cross geo writes that go over WANs.
- Last-Writer-Wins (LWW) Conflict Resolution: Some systems provide simplistic conflict resolution for all types of writes where the system clocks are used to determine the winner across conflicting writes. LWW is lightweight and can be suitable for simpler data. However, LWW can be destructive to updates that are not necessarily conflicting. For example adding a new element to a set across two geographies concurrently would result in only one of these new elements appearing in the final result with LWW.
- MVCC (multi-version concurrency control): MVCC systems maintain multiple versions of data and may expose ways for applications to resolve conflicts. Even though MVCC system can provide a flexible way to resolve conflicting writes, it comes at a cost of great complexity in the development of a solution.

Even though types and commands in Active-Active databases look identical to standard Redis types and commands, the underlying types in RS are enhanced to maintain more metadata to create the conflict-free data type experience. This section explains what you need to know about developing with Active-Active databases on Redis Enterprise Software.

Lua scripts

Active-Active databases support Lua scripts, but unlike standard Redis, Lua scripts always execute in effects replication mode. There is currently no way to execute them in script-replication mode.

Eviction

The default policy for Active-Active databases is `noeviction` mode. Redis Enterprise version 6.0.20 and later support all eviction policies for Active-Active databases, unless [Auto Tiering](#)(previously known as Redis on Flash) is enabled. For details, see [eviction for Active-Active databases](#).

Expiration

Expiration is supported with special multi-master semantics.

If a key's expiration time is changed at the same time on different members of the Active-Active database, the longer extended time set via TTL on a key is preserved. As an example:

If this command was performed on key1 on cluster #1

```
127.0.0.1:6379> EXPIRE key1 10
```

And if this command was performed on key1 on cluster #2

```
127.0.0.1:6379> EXPIRE key1 50
```

The EXPIRE command setting the key to 50 would win.

And if this command was performed on key1 on cluster #3:

```
127.0.0.1:6379> PERSIST key1
```

It would win out of the three clusters hosting the Active-Active database as it sets the TTL on key1 to an infinite time.

The replica responsible for the “winning” expire value is also responsible to expire the key and propagate a DEL effect when this happens. A “losing” replica is from this point on not responsible for expiring the key, unless another EXPIRE command resets the TTL. Furthermore, a replica that is NOT the “owner” of the expired value:

- Silently ignores the key if a user attempts to access it in READ mode, e.g. treating it as if it was expired but not propagating a DEL.
- Expires it (sending a DEL) before making any modifications if a user attempts to access it in WRITE mode.

 | **Note:** Expiration values are in the range of [0, 2^49] for Active-Active databases and [0, 2^64] for non Active-Active databases.

Out-of-Memory (OOM)

If a member Active-Active database is in an out of memory situation, that member is marked “inconsistent” by RS, the member stops responding to user traffic, and the syncer initiates full reconciliation with other peers in the Active-Active database.

Active-Active Database Key Counts

Keys are counted differently for Active-Active databases:

- DBSIZE (in shard-cli dbsize) reports key header instances that represent multiple potential values of a key before a replication conflict is resolved.
- expired_keys (in bdb-cli info) can be more than the keys count in DBSIZE (in shard-cli dbsize) because expires are not always removed when a key becomes a tombstone. A tombstone is a key that is logically deleted but still takes memory until it is collected by the garbage collector.
- The Expires average TTL (in bdb-cli info) is computed for local expires only.

INFO

The INFO command has an additional crdt section which provides advanced troubleshooting information (applicable to support etc.):

Section	Field	Description
CRDT Context	crdt_config_version	Currently active Active-Active database configuration version.
	crdt_slots	Hash slots assigned and reported by this shard.
	crdt_replid	Unique Replica/Shard IDs.
	crdt_clock	Clock value of local vector clock.
	crdt_ovc	Locally observed Active-Active database vector clock.
Peers	A list of currently connected Peer Replication peers. This is similar to the slaves list reported by Redis.	
Backlogs	A list of Peer Replication backlogs currently maintained. Typically in a full mesh topology only a single backlog is used for all peers, as the requested IDs are identical.	
CRDT Stats	crdt_sync_full	Number of inbound full synchronization processes performed.
	crdt_sync_partial_ok	Number of partial (backlog based) re-synchronization processes performed.
	crdt_sync_partial-err	Number of partial re-synchronization processes failed due to exhausted backlog.
	crdt_merge_reqs	Number of inbound merge requests processed.

Section	Field	Description
CRDT Replicas	crdt_effect_reqs	Number of inbound effect requests processed.
	crdt_ovc_filtered_effect_reqs	Number of inbound effect requests filtered due to old vector clock.
	crdt_gc_pending	Number of elements pending garbage collection.
	crdt_gc_attempted	Number of attempts to garbage collect tombstones.
	crdt_gc_collected	Number of tombstones garbaged collected successfully.
	crdt_gc_gvc_min	The minimal globally observed vector clock, as computed locally from all received observed clocks.
	crdt_stale_released_with_merge	Indicates last stale flag transition was a result of a complete full sync.
	A list of crdt_replica <uid> entries, each describes the known state of a remote instance with the following fields:	
	config_version	Last configuration version reported.
	shards	Number of shards.
	slots	Total number of hash slots.
	slot_coverage	A flag indicating remote shards provide full coverage (i.e. all shards are alive).
	max_ops_lag	Number of local operations not yet observed by the least updated remote shard
	min_ops_lag	Number of local operations not yet observed by the most updated remote shard

Updated: August 17, 2023

Data types for Active-Active databases

Active-Active databases use conflict-free replicated data types (CRDTs). From a developer perspective, most supported data types work the same for Active-Active and standard Redis databases. However, a few methods also come with specific requirements in Active-Active databases.

Even though they look identical to standard Redis data types, there are specific rules that govern the handling of conflicting concurrent writes for each data type.

As conflict handling rules differ between data types, some commands have slightly different requirements in Active-Active databases versus standard Redis databases.

See the following articles for more information

Updated: August 11, 2022

Hashes in an Active-Active databases

Hashes are great for structured data that contain a map of fields and values. They are used for managing distributed user or app session state, user preferences, form data and so on. Hash fields contain string type and string types operate just like the standard Redis string types when it comes to CRDTs. Fields in hashes can be initialized as a string using HSET or HMSET or can be used to initialize counter types that are numeric integers using HINCRBY or floats using HINCRBYFLOAT.

Hashes in Active-Active databases behave the same and maintain additional metadata to achieve an “OR-Set” behavior to handle concurrent conflicting writes. With the OR-Set behavior, writes to add new fields across multiple Active-Active database instances are typically unioned except in cases of conflicts. Conflicting instance writes can happen when an Active-Active database instance deletes a field while the other adds the same field. In this case and observed remove rule is followed. That is, remove can only remove fields it has already seen and in all other cases element add/update wins.

Field values behave just like CRDT strings. String values can be types string, counter integer based on the command used for initialization of the field value. See “String Data Type in Active-Active databases” and “String Data Type with Counter Value in Active-Active databases” for more details.

Here is an example of an “add wins” case:

Time	CRDB Instance1	CRDB Instance2
t1	HSET key1 field1 "a"	
t2		HSET key1 field2 "b"
t4	- Sync -	- Sync -
t5	HGETALL key1 1) "field2" 2) "b" 3) "field1" 4) "a"	HGETALL key1 1) "field2" 2) "b" 3) "field1" 4) "a"

Updated: August 11, 2022

HyperLogLog in Active-Active databases

HyperLogLog is an algorithm that addresses the [count-distinct problem](#). To do this it approximates the numbers of items in a set. Determining the exact cardinality of a set requires memory according to the cardinality of the set. Because it estimates the cardinality by probability, the HyperLogLog algorithm can run with more reasonable memory requirements.

HyperLogLog in Redis

Open source Redis implements [HyperLogLog](#) (HLL) as a native data-structure. It supports adding elements ([PFADD](#)) to an HLL, counting elements ([PFCOUNT](#)) of HLLs, and merging of ([PFMERGE](#)) HLLs.

Here is an example of a simple write case:

Time	Replica 1	Replica 2
t1	PFADD hll x	
t2	— sync —	
t3		PFADD hll y
t4	— sync —	
t5	PFCOUNT hll -> 2	PFCOUNT hll -> 2

Here is an example of a concurrent add case:

Time	Replica 1	Replica 2
t1	PFADD hll x	PFADD hll y
t2	PFCOUNT hll -> 1	PFCOUNT hll -> 1
t3	— sync —	
t4	PFCOUNT hll -> 2	PFCOUNT hll -> 2

The DEL-wins approach

Other collections in the Redis-CRDT implementation use the observed remove method to resolve conflicts. The CRDT-HLL uses the DEL-wins method. If a DEL request is received at the same time as any other request (ADD/MERGE/EXPIRE) on the HLL-key the replicas consistently converge to delete key. In the observed remove method used by other collections (sets, lists, sorted-sets and hashes), only the replica that received the DEL request removes the elements, but elements added concurrently in other replicas exist in the consistently converged collection. We chose to use the DEL-wins method for the CRDT-HLL to maintain the original time and space complexity of the HLL in open source Redis.

Here is an example of a DEL-wins case:

HLL		Set

HLL				Set		
Time	Replica 1	Replica 2		Time	Replica 1	Replica 2
t1	PFADD h e1			t1	SADD s e1	
t2	– sync –			t2	– sync –	
t3	PFCOUNT h -> 1	PFCOUNT h -> 1		t3	SCARD s -> 1	SCARD s -> 1
t4	PFADD h e2	Del h		t4	SADD s e2	Del S
t5	PFCOUNT h -> 2	PFCOUNT h -> 0		t5	SCARD s -> 2	SCARD s -> 0
t6	– sync –			t6	– sync –	
t7	PFCOUNT h -> 0	PFCOUNT h -> 0		t7	SCARD s -> 1	SCARD s -> 1
t8	Exists h -> 0	Exists h -> 0		t8	Exists s -> 1	Exists s -> 1
				t9	SMEMBERS s -> {e2} SMEMBERS s -> {e2}	

HLL in Active-Active databases versus HLL in Open Source Redis

In Active-Active databases, we implemented HLL within the CRDT on the basis of the Redis implementation with a few exceptions:

- Redis keeps the HLL data structure as an encoded string object such that you can potentially run any string request on a key that contains an HLL. In CRDT, only get and set are supported for HLL.
- In CRDT, if you do SET on a key that contains a value encoded as an HLL, then the value will remain an HLL. If the value is not encoded as HLL, then it will be a register.

Updated: August 11, 2022

JSON in Active-Active databases

Active-Active databases support JSON data structures.

To learn more about JSON in Active-Active databases, see [CRDT JSON command differences and conflict resolution rules](#).

Updated: August 17, 2023

Lists in Active-Active databases

Redis lists are simply lists of strings, sorted by insertion order. It is possible to add elements to a Redis List that push new elements to the head (on the left) or to the tail (on the right) of the list. Redis lists can be used to easily implement queues (using LPUSH and RPOP, for example) and stacks (using LPUSH and LPOP, for example).

Lists in Active-Active databases are just the same as regular Redis Lists. See the following examples to get familiar with Lists' behavior in an Active-Active database.

Simple Lists example:

	Time	CRDB Instance 1	CRDB Instance 2
	t1	LPUSH mylist "hello"	
	t2	– Sync –	– Sync –
	t3		LPUSH mylist "world"
	t4	– Sync –	– Sync –
	t5	LRANGE mylist 0 -1 =>"world" "hello"	LRANGE mylist 0 -1 => "world" "hello"

Explanation: The final list contains both the “world” and “hello” elements, in that order (Instance 2 observed “hello” when it added “world”).

Example of Lists with Concurrent Insertions:

Time	CRDB Instance 1	CRDB Instance 2
t1	LPUSH L x	
t2	– Sync –	– Sync –
t3	LINSERT L AFTER xy1	
t4		LINSERT L AFTER xy2
t5	LRANGE L 0 -1 => xy1	LRANGE L 0 -1 => xy2
t6	– Sync –	– Sync –
t7	LRANGE L 0 -1 => xy1 y2	LRANGE L 0 -1 => xy1 y2

Explanation: Instance 1 added an element y1 after x, and then Instance 2 added element y2 after x. The final List contains all three elements: x is the first element, after it y1 and then y2. The Active-Active database resolves the conflict arbitrarily but applies the resolution consistently across all Active-Active database instances.

Example of Deleting a List while Pushing a New Element:

Time	CRDB Instance 1	CRDB Instance 2
t1	LPUSH L x	
t2	– Sync –	– Sync –
t3	LRANGE L 0 -1 => x	LRANGE L 0 -1 => x
t4	LPUSH L y	DEL L
t5	– Sync –	– Sync –
t6	LRANGE L 0 -1 => y	LRANGE L 0 -1 => y

Explanation At t4 - t6, DEL deletes only observed elements. This is why L still contains y.

Example of Popping Elements from a List:

Time	CRDB Instance 1	CRDB Instance 2
t1	LPUSH L x y z	
t2	– Sync –	– Sync –
t3		RPOP L => x
t4	– Sync –	– Sync –
t5	RPOP L => y	
t6	– Sync –	– Sync –
t7	RPOP L => z	RPOP L => z

Explanation: At t1, the operation pushes elements x, y, z to List L. At t3, the sequential pops behave as expected from a queue. At t7, the concurrent pop in both instances might show the same result. The instance was not able to sync regarding the z removal so, from the point of view of each instance, z is located in the List and can be popped. After syncing, both lists are empty.

Be aware of the behavior of Lists in Active-Active databases when using List as a stack or queue. As seen in the above example, two parallel RPOP operations performed by two different Active-Active database instances can get the same element in the case of a concurrent operation. Lists in Active-Active databases guarantee that each element is POP-ed at least once, but cannot guarantee that each element is POP-ed only once. Such behavior should be taken into account when, for example, using Lists in Active-Active databases as building blocks for inter-process communication systems.

In that case, if the same element cannot be handled twice by the applications, it's recommended that the POP operations be performed by one Active-Active database instance, whereas the PUSH operations can be performed by multiple instances.

Updated: August 11, 2022

Sets in Active-Active databases

A Redis set is an unordered collection of strings. It is possible to add, remove, and test for the existence of members with Redis commands. A Redis set maintains a unique collection of elements. Sets can be great for maintaining a list of events (click streams), users (in a group conversation), products (in recommendation lists), engagements (likes, shares) and so on.

Sets in Active-Active databases behave the same and maintain additional metadata to achieve an “OR-Set” behavior to handle concurrent conflicting writes. With the OR-Set behavior, writes across multiple Active-Active database instances are typically unioned except in cases of conflicts. Conflicting instance writes can happen when a Active-Active database instance deletes an element while the other adds the same element. In this case an observed remove rule is followed. That is, remove can only remove instances it has already seen and in all other cases element add wins.

Here is an example of an “add wins” case:

Time	CRDB Instance1	CRDB Instance2
t1	SADD key1 “a”	
t2		SADD key1 “b”
t3	SMEMBERS key1 “a”	SMEMBERS key1 “b”
t4	– Sync –	– Sync –
t3	SMEMBERS key1 “a” “b”	SMEMBERS key1 “a” “b”

Here is an example of an “observed remove” case.

Time	CRDB Instance1	CRDB Instance2
t1	SMEMBERS key1 “a” “b”	SMEMBERS key1 “a” “b”
t2	SREM key1 “a”	SADD key1 “c”
t3	SREM key1 “c”	
t4	– Sync –	– Sync –
t3	SMEMBERS key1 “c” “b”	SMEMBERS key1 “c” “b”

Updated: August 11, 2022

Sorted sets in Active-Active databases

! Note: [Redis Geospatial \(Geo\)](#) is based on Sorted Sets, so the same Active-Active database development instructions apply to Geo.

Similar to Redis Sets, Redis Sorted Sets are non-repeating collections of Strings. The difference between the two is that every member of a Sorted Set is associated with a score used to order the Sorted Set from lowest to highest. While members are unique, they may have the same score.

With Sorted Sets, you can quickly add, remove or update elements as well as get ranges by score or by rank (position). Sorted Sets in Active-Active databases behave the same and maintain additional metadata to handle concurrent conflicting writes. Conflict resolution is done in two phases:

1. First, the database resolves conflict at the set level using “OR Set” (Observed-Remove Set). With OR-Set behavior, writes across multiple Active-Active database instances are typically unioned except in cases of conflicts. Conflicting writes can happen when an Active-Active database instance deletes an element while the other adds or updates the same element. In this case, an observed Remove rule is followed, and only instances it has already seen are removed. In all other cases, the Add / Update element wins.
2. Second, the database resolves conflict at the score level. In this case, the score is treated as a counter and applies the same conflict resolution as regular counters.

See the following examples to get familiar with Sorted Sets' behavior in Active-Active database:

Example of Simple Sorted Set with No Conflict:

Time	CRDB Instance 1	CRDB Instance 2
t1	ZADD Z 1.1 x	
t2	– Sync –	– Sync –
t3		ZADD Z 1.2 y
t4	– Sync –	– Sync –
t5	ZRANGE Z 0 -1 => x y	ZRANGE Z 0 -1 => x y

Explanation: When adding two different elements to a Sorted Set from different replicas (in this example, x with score 1.1 was added by Instance 1 to Sorted

Set Z, and y with score 1.2 was added by Instance 2 to Sorted Set Z) in a non-concurrent manner (i.e. each operation happened separately and after both instances were in sync), the end result is a Sorted Set including both elements in each Active-Active database instance. Example of Sorted Set and Concurrent Add:

Time	CRDB Instance 1	CRDB Instance 2
t1	ZADD Z 1.1 x	
t2		ZADD Z 2.1 x
t3	ZSCORE Z x => 1.1	ZSCORE Z x => 2.1
t4	– Sync –	– Sync –
t5	ZSCORE Z x => 2.1	ZSCORE Z x => 2.1

Explanation: When concurrently adding an element x to a Sorted Set Z by two different Active-Active database instances (Instance 1 added score 1.1 and Instance 2 added score 2.1), the Active-Active database implements Last Write Win (LWW) to determine the score of x. In this scenario, Instance 2 performed the ZADD operation at time t2>t1 and therefore the Active-Active database sets the score 2.1 to x.

Example of Sorted Set with Concurrent Add Happening at the Exact Same Time:

Time	CRDB Instance 1	CRDB Instance 2
t1	ZADD Z 1.1 x	ZADD Z 2.1 x
t2	ZSCORE Z x => 1.1	ZSCORE Z x => 2.1
t3	– Sync –	– Sync –
t4	ZSCORE Z x => 1.1	ZSCORE Z x => 1.1

Explanation: The example above shows a relatively rare situation, in which two Active-Active database instances concurrently added the same element x to a Sorted Set at the same exact time but with a different score, i.e. Instance 1 added x with a 1.1 score and Instance 2 added x with a 2.1 score. After syncing, the Active-Active database realized that both operations happened at the same time and resolved the conflict by arbitrarily (but consistently across all Active-Active database instances) giving precedence to Instance 1. Example of Sorted Set with Concurrent Counter Increment:

Time	CRDB Instance 1	CRDB Instance 2
t1	ZADD Z 1.1 x	
t2	– Sync –	– Sync –
t3	ZINCRBY Z 1.0 x	ZINCRBY Z 1.0 x
t4	– Sync –	– Sync –
t5	ZSCORE Z x => 3.1	ZSCORE Z x => 3.1

Explanation: The result is the sum of all ZINCRBY operations performed by all Active-Active database instances.

Example of Removing an Element from a Sorted Set:

Time	CRDB Instance 1	CRDB Instance 2
t1	ZADD Z 4.1 x	
t2	– Sync –	– Sync –
t3	ZSCORE Z x => 4.1	ZSCORE Z x => 4.1
t4	ZREM Z x	ZINCRBY Z 2.0 x
t5	ZSCORE Z x => nill	ZSCORE Z x => 6.1
t6	– Sync –	– Sync –
t7	ZSCORE Z x => 2.0	ZSCORE Z x => 2.0

Explanation: At t4 - t5, concurrent ZREM and ZINCRBY operations ran on Instance 1 and Instance 2 respectively. Before the instances were in sync, the ZREM operation could only delete what had been seen by Instance 1, so Instance 2 was not affected. Therefore, the ZSCORE operation shows the local effect on x. At t7, after both instances were in-sync, the Active-Active database resolved the conflict by subtracting 4.1 (the value of element x in Instance 1) from 6.1 (the value of element x in Instance 2).

Streams in Active-Active databases

A Redis Stream is a data structure that acts like an append-only log. Each stream entry consists of:

- A unique, monotonically increasing ID
- A payload consisting of a series key-value pairs

You add entries to a stream with the XADD command. You access stream entries using the XRANGE, XREADGROUP, and XREAD commands (however, see the caveat about XREAD below).

Streams and Active-Active

Active-Active databases allow you to write to the same logical stream from more than one region. Streams are synchronized across the regions of an Active-Active database.

In the example below, we write to a stream concurrently from two regions. Notice that after syncing, both regions have identical streams:

Time	Region 1	Region 2
t1	XADD messages * text hello	XADD messages * text goodbye
t2	XRANGE messages - + →[1589929244828-1]	XRANGE messages - + →[1589929246795-2]
t3	– Sync –	– Sync –
t4	XRANGE messages - + →[1589929244828-1, 1589929246795-2]	XRANGE messages - + →[1589929244828-1, 1589929246795-2]

Notice also that the synchronized streams contain no duplicate IDs. As long as you allow the database to generate your stream IDs, you'll never have more than one stream entry with the same ID.



Note: Open source Redis uses one radix tree (referred to as `rax` in the code base) to implement each stream. However, Active-Active databases implement a single logical stream using one `rax` per region. Each region adds entries only to its associated `rax` (but can remove entries from all `rax` trees). This means that XREAD and XREADGROUP iterate simultaneously over all `rax` trees and return the appropriate entry by comparing the entry IDs from each `rax`.

Conflict resolution

Active-Active databases use an “observed-remove” approach to automatically resolve potential conflicts.

With this approach, a delete only affects the locally observable data.

In the example below, a stream, x, is created at t1. At t3, the stream exists in two regions.

Time	Region 1	Region 2
t1	XADD messages * text hello	
t2	– Sync –	– Sync –
t3	XRANGE messages - + →[1589929244828-1]	XRANGE messages - + →[1589929244828-1]
t4	DEL messages	XADD messages * text goodbye
t5	– Sync –	– Sync –
t6	XRANGE messages - + →[1589929246795-2]	XRANGE messages - + →[1589929246795-2]

At t4, the stream is deleted from Region 1. At the same time, an entry with ID ending in 3700 is added to the same stream at Region 2. After the sync, at t6, the entry with ID ending in 3700 exists in both regions. This is because that entry was not visible when the local stream was deleted at t4.

ID generation modes

Usually, you should allow Redis streams generate its own stream entry IDs. You do this by specifying * as the ID in calls to XADD. However, you can provide your own custom ID when adding entries to a stream.

Because Active-Active databases replicate asynchronously, providing your own IDs can create streams with duplicate IDs. This can occur when your write to

the same stream from multiple regions.

Time	Region 1	Region 2
t1	XADD x 100-1 f1 v1	XADD x 100-1 f1 v1
t2	– Sync –	– Sync –
t3	XRANGE x - + → [100-1, 100-1]	XRANGE x - + → [100-1, 100-1]

In this scenario, two entries with the ID 100-1 are added at t1. After syncing, the stream x contains two entries with the same ID.

 **Note:** Stream IDs in open source Redis consist of two integers separated by a dash ('-'). When the server generates the ID, the first integer is the current time in milliseconds, and the second integer is a sequence number. So, the format for stream IDs is MS-SEQ.

To prevent duplicate IDs and to comply with the original Redis streams design, Active-Active databases provide three ID modes for XADD:

- Strict:** In strict mode, XADD allows server-generated IDs (using the '*' ID specifier) or IDs consisting only of the millisecond (MS) portion. When the millisecond portion of the ID is provided, the ID's sequence number is calculated using the database's region ID. This prevents duplicate IDs in the stream. Strict mode rejects full IDs (that is, IDs containing both milliseconds and a sequence number).
- Semi-strict:** Semi-strict mode is just like strict mode except that it allows full IDs (MS-SEQ). Because it allows full IDs, duplicate IDs are possible in this mode.
- Liberal:** XADD allows any monotonically ascending ID. When given the millisecond portion of the ID, the sequence number will be set to 0. This mode may also lead to duplicate IDs.

The default and recommended mode is strict, which prevents duplicate IDs

 **Warning -** Why do you want to prevent duplicate IDs? First, XDEL, XCLAIM, and other commands can affect more than one entry when duplicate IDs are present in a stream. Second, duplicate entries may be removed if a database is exported or renamed.

To change XADD's ID generation mode, use the rladmin command-line utility:

Set strict mode:

```
rladmin tune db crdb crdt_xadd_id_uniqueness_mode strict
```

Set semi-strict mode:

```
rladmin tune db crdb crdt_xadd_id_uniqueness_mode semi-strict
```

Set liberal mode:

```
rladmin tune db crdb crdt_xadd_id_uniqueness_mode liberal
```

Iterating a stream with XREAD

In open source Redis and in non-Active-Active databases, you can use XREAD to iterate over the entries in a Redis Stream. However, with an Active-Active database, XREAD may skip entries. This can happen when multiple regions write to the same stream.

In the example below, XREAD skips entry 115-2.

Time	Region 1	Region 2
t1	XADD x 110 f1 v1	XADD x 115 f1 v1
t2	XADD x 120 f1 v1	
t3	XADD x 130 f1 v1	
t4	XREAD COUNT 2 STREAMS x 0 → [110-1, 120-1]	
t5	– Sync –	– Sync –
t6	XREAD COUNT 2 STREAMS x 120-1 → [130-1]	
t7	XREAD STREAMS x 0 → [110-1, 115-2, 120-1, 130-1]	XREAD STREAMS x 0 → [110-1, 115-2, 120-1, 130-1]

You can use XREAD to reliably consume a stream only if all writes to the stream originate from a single region. Otherwise, you should use XREADGROUP, which always guarantees reliable stream consumption.

Consumer groups

Active-Active databases fully support consumer groups with Redis Streams. Here is an example of creating two consumer groups concurrently:

Time	Region 1	Region 2
t1	XGROUP CREATE x group1 0	XGROUP CREATE x group2 0
t2	XINFO GROUPS x → [group1]	XINFO GROUPS x → [group2]
t3	– Sync –	– Sync –
t4	XINFO GROUPS x → [group1, group2]	XINFO GROUPS x → [group1, group2]



Note:

Open source Redis uses one radix tree (`rax`) to hold the global pending entries list and another `rax` for each consumer's PEL. The global PEL is a unification of all consumer PELs, which are disjoint.

An Active-Active database stream maintains a global PEL and a per-consumer PEL for each region.

When given an ID different from the special ">" ID, XREADGROUP iterates simultaneously over all of the PELs for all consumers. It returns the next entry by comparing entry IDs from the different PELs.

Conflict resolution

The “delete wins” approach is a way to automatically resolve conflicts with consumer groups. In case of concurrent consumer group operations, a delete will “win” over other concurrent operations on the same group.

In this example, the DEL at t4 deletes both the observed group1 and the non-observed group2:

Time	Region 1	Region 2
t1	XGROUP CREATE x group1 0	
t2	– Sync –	– Sync –
t3	XINFO GROUPS x → [group1]	XINFO GROUPS x → [group1]
t4	DEL x	XGROUP CREATE x group2 0
t5	– Sync –	– Sync –
t6	EXISTS x → 0	EXISTS x → 0

In this example, the XGROUP DESTROY at t4 affects both the observed group1 created in Region 1 and the non-observed group1 created in Region 3:

time	Region 1	Region 2	Region 3
t1	XGROUP CREATE x group1 0		
t2	– Sync –	– Sync –	
t3	XINFO GROUPS x → [group1]	XINFO GROUPS x → [group1]	XINFO GROUPS x → []
t4		XGROUP DESTROY x group1	XGROUP CREATE x group1 0
t5	– Sync –	– Sync –	– Sync –
t6	EXISTS x → 0	EXISTS x → 0	EXISTS x → 0

Group replication

Calls to XREADGROUP and XACK change the state of a consumer group or consumer. However, it's not efficient to replicate every change to a consumer or consumer group.

To maintain consumer groups in Active-Active databases with optimal performance:

1. Group existence (CREATE/DESTROY) is replicated.
2. Most XACK operations are replicated.
3. Other operations, such as XGROUP, SETID, DELCONSUMER, are not replicated.

For example:

Time	Region 1	Region 2
t1	XADD messages 110 text hello	
t2	XGROUP CREATE messages group1 0	
t3	XREADGROUP GROUP group1 Alice STREAMS messages > → [110-1]	
t4	– Sync –	– Sync –
t5	XRANGE messages - + → [110-1]	XRANGE messages - + → [110-1]
t6	XINFO GROUPS messages → [group1]	XINFO GROUPS messages → [group1]
t7	XINFO CONSUMERS messages group1 → [Alice]	XINFO CONSUMERS messages group1 → []
t8	XPENDING messages group1 - + 1 → [110-1]	XPENDING messages group1 - + 1 → []

Using XREADGROUP across regions can result in regions reading the same entries. This is due to the fact that Active-Active Streams is designed for at-least-once reads or a single consumer. As shown in the previous example, Region 2 is not aware of any consumer group activity, so redirecting the XREADGROUP traffic from Region 1 to Region 2 results in reading entries that have already been read.

Replication performance optimizations

Consumers acknowledge messages using the XACK command. Each ack effectively records the last consumed message. This can result in a lot of cross-region traffic. To reduce this traffic, we replicate XACK messages only when all of the read entries are acknowledged.

Time	Region 1	Region 2	Explanation
t1	XADD x 110-0 f1 v1		
t2	XADD x 120-0 f1 v1		
t3	XADD x 130-0 f1 v1		
t4	XGROUP CREATE x group1 0		
t5	XREADGROUP GROUP group1 Alice STREAMS x > → [110-0, 120-0, 130-0]		
t6	XACK x group1 110-0		
t7	– Sync –	– Sync –	110-0 and its preceding entries (none) were acknowledged. We replicate an XACK effect for 110-0.
t8	XACK x group1 130-0		130-0 was acknowledged, but not its preceding entries (120-0). We DO NOT replicate an XACK effect for 130-0
t9	– Sync –	– Sync –	
t10	XACK x group1 120-0		120-0 and its preceding entries (110-0 through 130-0) were acknowledged. We replicate an XACK effect for 130-0.
t11	– Sync –	– Sync –	

In this scenario, if we redirect the XREADGROUP traffic from Region 1 to Region 2 we do not re-read entries 110-0, 120-0 and 130-0. This means that the XREADGROUP does not return already-acknowledged entries.

Guarantees

Unlike XREAD, XREADGROUP will never skip stream entries. In traffic redirection, XREADGROUP may return entries that have been read but not acknowledged. It may also even return entries that have already been acknowledged.

Summary

With Active-Active streams, you can write to the same logical stream from multiple regions. As a result, the behavior of Active-Active streams differs somewhat from the behavior you get with open source Redis. This is summarized below:

Stream commands

1. When using the strict ID generation mode, XADD does not permit full stream entry IDs (that is, an ID containing both MS and SEQ).
2. XREAD may skip entries when iterating a stream that is concurrently written to from more than one region. For reliable stream iteration, use XREADGROUP instead.
3. XSETID fails when the new ID is less than current ID.

Consumer group notes

The following consumer group operations are replicated:

1. Consecutive XACK operations
2. Consumer group creation and deletion (that is, XGROUP CREATE and XGROUP DESTROY)

All other consumer group metadata is not replicated.

A few other notes:

1. XGROUP SETID and DELCONSUMER are not replicated.
2. Consumers exist locally (XREADGROUP creates a consumer implicitly).
3. Renaming a stream (using RENAME) deletes all consumer group information.

Updated: August 17, 2023

Strings and bitfields in Active-Active databases

Active-Active databases support both strings and bitfields.



Note: Active-Active **bitfield** support was added in RS version 6.0.20.

Changes to both of these data structures will be replicated across Active-Active member databases.

Replication semantics

Except in the case of [string counters](#) (see below), both strings and bitfields are replicated using a “last write wins” approach. The reason for this is that strings and bitfields are effectively binary objects. So, unlike with lists, sets, and hashes, the conflict resolution semantics of a given operation on a string or bitfield are undefined.

How “last write wins” works

A wall-clock timestamp (OS time) is stored in the metadata of every string and bitfield operation. If the replication syncer cannot determine the order of operations, the value with the latest timestamp wins. This is the only case with Active-Active databases where OS time is used to resolve a conflict.

Here’s an example where an update happening to the same key at a later time (t2) wins over the update at t1.

Time	Region 1	Region 2
t1	SET text “a”	
t2		SET text “b”
t3	– Sync –	– Sync –
t4	SET text “c”	
t5	– Sync –	– Sync –
t6		SET text “d”

String counter support

When you're using a string as counter (for instance, with the [INCR](#) or [INCRBY](#) commands), then conflicts will be resolved semantically.

On conflicting writes, counters accumulate the total counter operations across all member Active-Active databases in each sync.

Here's an example of how counter values works when synced between two member Active-Active databases. With each sync, the counter value accumulates the private increment and decrements of each site and maintain an accurate counter across concurrent writes.

Time	Region 1	Region 2
t1	INCRBY counter 7	
t2		INCRBY counter 3
t3	GET counter 7	GET counter 3
t4	— Sync —	— Sync —
t5	GET counter 10	GET counter 10
t6	DECRBY counter 3	
t7		INCRBY counter 6
t8	— Sync —	— Sync —
t9	GET counter 13	GET counter 13

 Note: Active-Active databases support 59-bit counters. This limitation is to protect from overflowing a counter in a concurrent operation.

Updated: August 11, 2022

Application failover with Active-Active databases

Active-Active Redis deployments don't have a built-in failover or fallback mechanism for application connections. An application deployed with an Active-Active database connects to a replica of the database that is geographically nearby. If that replica is not available, the application can failover to a remote replica, and fallback again if necessary. In this article we explain how this process works.

Active-Active connection failover can improve data availability, but can negatively impact data consistency. Active-Active replication, like Redis replication, is asynchronous. An application that fails over to another replica can miss write operations. If the failed replica saved the write operations in persistent storage, then the write operations are processed when the failed replica recovers.

Detecting Failure

Your application can detect two types of failure:

1. Local failures - The local replica is down or otherwise unavailable
2. Replication failures - The local replica is available but fails to replicate to or from remote replicas

Local Failures

Local failure is detected when the application is unable to connect to the database endpoint for any reason. Reasons for a local failure can include: multiple node failures, configuration errors, connection refused, connection timed out, unexpected protocol level errors.

Replication Failures

Replication failures are more difficult to detect reliably without causing false positives. Replication failures can include: network split, replication configuration issues, remote replica failures.

The most reliable method for health-checking replication is by using the Redis publish/subscribe (pub/sub) mechanism.

 Note: Note that this document does not suggest that Redis pub/sub is reliable in the common sense. Messages can get lost in certain conditions, but that is acceptable in this case because typically the application determines that replication is down only after not being able to deliver a number of messages over a period of time.

When you use the pub/sub data type to detect failures, the application:

1. Connects to all replicas and subscribes to a dedicated channel for each replica.
2. Connects to all replicas and periodically publishes a uniquely identifiable message.
3. Monitors received messages and ensures that it is able to receive its own messages within a predetermined window of time.

You can also use known dataset changes to monitor the reliability of the replication stream, but pub/sub is preferred method because:

1. It does not involve dataset changes.
2. It does not make any assumptions about the dataset.
3. Pub/sub messages are delivered as replicated effects and are a more reliable indicator of a live replication link. In certain cases, dataset keys may appear to be modified even if the replication link fails. This happens because keys may receive updates through full-state replication (re-sync) or through online replication of effects.

Impact of sharding on failure detection

If your sharding configuration is symmetric, make sure to use at least one key (PUB/SUB channels or real dataset key) per shard. Shards are replicated individually and are vulnerable to failure. Symmetric sharding configurations have the same number of shards and hash slots for all replicas. We do not recommend an asymmetric sharding configuration, which requires at least one key per hash slot that intersects with a pair of shards.

To make sure that there is at least one key per shard, the application should:

1. Use the Cluster API to retrieve the database sharding configuration.
2. Compute a number of key names, such that there is one key per shard.
3. Use those key names as channel names for the pub/sub mechanism.

Failing over

When the application needs to failover to another replica, it should simply re-establish its connections with the endpoint on the remote replica. Because Active/Active and Redis replication are asynchronous, the remote endpoint may not have all of the locally performed and acknowledged writes.

It's best if your application doesn't read its own recent writes. Those writes can be either:

1. Lost forever, if the local replica has an event such as a double failure or loss of persistent files.
2. Temporarily unavailable, but will be available at a later time if the local replica's failure is temporary.

Failback decision

Your application can use the same checks described above to continue monitoring the state of the failed replica after failover.

To monitor the state of a replica during the failback process, you must make sure the replica is available, re-synced with the remote replicas, and not in stale mode. The PUB/SUB mechanism is an effective way to monitor this.

Dataset-based mechanisms are potentially less reliable for several reasons:

1. In order to determine that a local replica is not stale, it is not enough to simply read keys from it. You must also attempt to write to it.
2. As stated above, remote writes for some keys appear in the local replica before the replication link is back up and while the replica is still in stale mode.
3. A replica that was never written to never becomes stale, so on startup it is immediately ready but serves stale data for a longer period of time.

Replica Configuration Changes

All failover and failback operations should be done strictly on the application side, and should not involve changes to the Active-Active configuration. The only valid case for re-configuring the Active-Active deployment and removing a replica is when memory consumption becomes too high because garbage collection cannot be performed. Once a replica is removed, it can only be re-joined as a new replica and it loses any writes that were not converged.

Updated: June 6, 2022

Auto Tiering

Redis Enterprise's auto tiering offers users the unique ability to use solid state drives (SSDs) to extend databases beyond DRAM capacity. Developers can build applications that require large datasets using the same Redis API. Using SSDs can significantly reduce the infrastructure costs compared to only DRAM deployments.

Frequently used data, called hot data, belongs in the fastest memory level to deliver a real-time user experience. Data that is accessed less frequently, called warm data, can be kept in a slightly slower memory tier. Redis Enterprise's Auto tiering maintains hot data in DRAM, keeps warm data in SSDs, and transfers data between tiers automatically.

Redis Enterprise's auto tiering is based on a high-performance storage engine (Speedb) that manages the complexity of using SSDs and DRAM as the total available memory for databases in a Redis Enterprise cluster. This implementation offers a performance boost of up to 10k operations per second per core of the database, doubling the performance of Redis on Flash.

Just like all-RAM databases, databases with Auto Tiering enabled are compatible with existing Redis applications.

Auto Tiering is also supported on [Redis Cloud](#) and [Redis Enterprise Software for Kubernetes](#).

Use cases

The benefits associated with Auto Tiering are dependent on the use case.

Auto Tiering is ideal when your:

- working set is significantly smaller than your dataset (high RAM hit rate)
- average key size is smaller than average value size (all key names are stored in RAM)
- most recent data is the most frequently used (high RAM hit rate)

Auto Tiering is not recommended for:

- Long key names (all key names are stored in RAM)
- Broad access patterns (any value could be pulled into RAM)
- Large working sets (working set is stored in RAM)
- Frequently moved data (moving to and from RAM too often can impact performance)

Auto Tiering is not intended to be used for persistent storage. Redis Enterprise Software database persistent and ephemeral storage should be on different disks, either local or attached.

Where is my data?

When using Auto Tiering, RAM storage holds:

- All keys (names)
- Key indexes
- Dictionaries
- Hot data (working set)

All data is accessed through RAM. If a value in flash memory is accessed, it becomes part of the working set and is moved to RAM. These values are referred to as "hot data".

Inactive or infrequently accessed data is referred to as "warm data" and stored in flash memory. When more space is needed in RAM, warm data is moved from RAM to flash storage.



Note: When using Auto Tiering with RediSearch, it's important to note that RediSearch indexes are also stored in RAM.

RAM to Flash ratio

Redis Enterprise Software allows you to configure and tune the ratio of RAM-to-Flash for each database independently, optimizing performance for your specific use case. While this is an online operation requiring no downtime for your database, it is recommended to perform it during maintenance windows as data might move between tiers (RAM <-> Flash).

The RAM size cannot be smaller than 10% or larger than 60% of the total memory. We recommend you keep at least 20% of all values in RAM.

Flash memory

Implementing Auto Tiering requires pre planning around memory and sizing. Considerations and requirements for Auto Tiering include:

- Flash memory must be locally attached (as opposed to network attached storage (NAS) and storage area networks (SAN)).
- Flash memory must be dedicated to Auto Tiering and not shared with other parts of the database, such as durability, binaries, or persistence.
- For the best performance, the SSDs should be NVMe based, but SATA can also be used.



Note: The Redis Enterprise Software database persistent and ephemeral storage should be on different disks, either local or attached.

Once these requirements are met, you can create and manage both databases with Auto Tiering enabled and all-RAM databases in the same cluster.

When you begin planning the deployment of an Auto Tiering enabled database in production, we recommend working closely with the Redis technical team

for sizing and performance tuning.

Cloud environments

When running in a cloud environment:

- Flash memory is on the ephemeral SSDs of the cloud instance (for example the local NVMe of AWS i4i instances and Azure Lsv2 and Lsv3 series).
- Persistent database storage needs to be network attached (for example, AWS EBS for AWS).

 **Note:** We specifically recommend “[Storage Optimized I4i - High I/O Instances](#)” because of the performance of NVMe for flash memory.

On-premises environments

When you begin planning the deployment of Auto Tiering in production, we recommend working closely with the Redis technical team for sizing and performance tuning.

On-premises environments support more deployment options than other environments such as:

- Using Redis Stack features:
 - [Search and query](#)
 - [JSON](#)
 - [Time series](#)
 - [Probabilistic data structures](#)

 **Note:** Enabling Auto Tiering for Active-Active distributed databases requires validating and getting the Redis technical team’s approval first.

 **Warning** - Auto Tiering is not supported running on network attached storage (NAS), storage area network (SAN), or with local HDD drives.

Next steps

- [Auto Tiering metrics](#)
- [Auto Tiering quick start](#)
- [Ephemeral and persistent storage](#)
- [Hardware requirements](#)

Updated: August 17, 2023

Auto Tiering quick start

This page guides you through a quick setup of [Auto Tiering](#) with a single node for testing and demo purposes.

For production environments, you can find more detailed installation instructions in the [install and setup](#) section.

The steps to set up a Redis Enterprise Software cluster using Auto Tiering with a single node are:

1. Install Redis Enterprise Software or run it in a Docker container.
2. Set up a Redis Enterprise Software cluster with Auto Tiering.
3. Create a new database with Auto Tiering enabled.
4. Connect to your new database.

Install Redis Enterprise Software

Bare metal, VM, Cloud instance

To install on bare metal, a virtual machine, or an instance:

1. Download the binaries from the [Redis Enterprise download center](#).

2. Upload the binaries to a Linux-based operating system.

3. Extract the image:

```
tar -vxf <downloaded tar file name>
```

4. After the `tar` command completes, you can find a new `install.sh` script in the current directory:

```
sudo ./install.sh -y
```

Docker-based installation

For testing purposes, you can run a Redis Enterprise Software Docker container on Windows, MacOS, and Linux.

```
docker run -d --cap-add sys_resource --name rp -p 8443:8443 -p 12000:12000 redislabs/redis:latest
```

Set up a cluster and enable Auto Tiering

1. Direct your browser to `https://localhost:8443/new` on the host machine to see the Redis Enterprise Software Cluster Manager UI.



Note: Depending on your browser, you may see a certificate error. Choose “continue to the website” to go to the setup screen.

2. Select **Create new cluster**.

3. Set up account credentials for a cluster administrator, then select **Next** to proceed to cluster setup.

4. Enter your cluster license key if you have one. Otherwise, the cluster uses the trial version.

5. Provide a cluster FQDN such as `mycluster.local`, then select **Next**.

6. In the **Storage configuration** section, turn on the **Enable flash storage** toggle.

7. Select **Create cluster**.

8. Select **OK** to confirm that you are aware of the replacement of the HTTPS TLS certificate on the node, and proceed through the browser warning.

Create a database

On the **Databases** screen:

1. Select **Quick database**.

2. Verify **Flash** is selected for **Runs on**.

Quick Database

Cancel

Create

Runs on

Flash

RAM

Database name

Database-25-Jul-23-15-54PM

Port

12000

Will be set by the cluster
if not set manually

Memory limit (GB)

1



2.22 GB RAM , 17.35 GB flash available

RAM limit (GB)

0.1

/ 1GB

10%

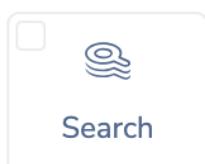
10%

100%

Recommended up to 0.50GB (50%)

Modules

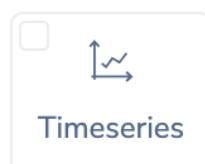
Your selection cannot be changed at a later stage



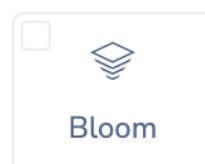
Search



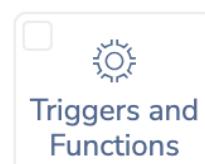
JSON



Timeseries



Bloom



Triggers and
Functions

🔗 Modules parameters

ⓘ Use full option for high availability and persistence

Full options

3. Enter 12000 for the endpoint Port number.
4. (Optional) Select Full options to see available alerts.
5. Select Create.

You now have a database with Auto Tiering enabled!

Connect to your database

You are ready to connect to your database to store data. See the [test connectivity](#) page to learn how to connect to your database.

Next steps

If you want to generate load against the database or add a bunch of data for cluster testing, see the [memtier_benchmark quick start](#) for help.

To see the true performance and scale of Auto Tiering, you must tune your I/O path and set the flash path to the mounted path of SSD or NVMe flash memory as that is what it is designed to run on. For more information, see [Auto Tiering](#).

Updated: August 17, 2023

Manage Auto Tiering storage engine

Manage the storage engine

Redis Enterprise Auto Tiering supports two storage engines:

- [Speedb](#) (default, recommended)
- [RocksDB](#)



Warning - Switching between storage engines requires guidance by Redis Support or your Account Manager.

Change the storage engine

1. Change the cluster level configuration for default storage engine.

- API:

```
curl -k -u <username>:<password> -X PUT -H "Content-Type: application/json" -d
'{"bigstore_driver": "speedb"}' https://localhost:9443/v1/cluster
```

- CLI:

```
rladmin cluster config bigstore_driver {speedb | rocksdb}
```

2. Restart the each database on the cluster one by one.

```
rladmin restart db { db:<id> | <name> }
```



Note: We recommend restarting your database at times with low usage and avoiding peak hours. For databases without persistence enabled, we also recommend using export to backup your database first.

Monitor the storage engine

To get the current cluster level default storage engine run:

- Use the `rladmin info cluster` command look for 'bigstore_driver'.
- Use the REST API:

```
curl -k -u <username>:<password> -X GET -H "Content-Type: application/json"
https://localhost:9443/v1/cluster
```

Versions of Redis Enterprise 7.2 and later provide a metric called `bdb_bigstore_shard_count` to help track the shard count per database, filtered by `bdb_id` and by storage engine as shown below:

```
bdb_bigstore_shard_count{bdb="1",cluster="mycluster.local",driver="rocksdb"} 1.0  
bdb_bigstore_shard_count{bdb="1",cluster="mycluster.local",driver="speedb"} 2.0
```

For more about metrics for Redis Enterprise's integration with Prometheus, see [Prometheus integration](#).

Updated: August 17, 2023

Durability and high availability

Redis Enterprise Software comes with several features that make your data more durable and accessible. The following features can help protect your data in cases of failures or outages and help keep your data available when you need it.

Replication

When you replicate your database, each database instance (shard) is copied one or more times. Your database will have one primary shard and one or more replica shards. When a primary shard fails, Redis Enterprise automatically promotes a replica shard to primary.

Clustering

Clustering (or sharding) breaks your database into individual instances (shards) and spreads them across several nodes. Clustering lets you add resources to your cluster to scale your database and prevents node failures from causing availability loss.

Database persistence

Database persistence gives your database durability against process or server failures by saving data to disk at set intervals.

Active-Active geo-distributed replication

Active-Active Redis Enterprise databases distribute your replicated data across multiple nodes and availability zones. This increases the durability of your database by reducing the likelihood of data or availability loss. It also reduces data access latency.

Rack-zone awareness

Rack-zone awareness maps each node in your Redis Enterprise cluster to a physical rack or logical zone. The cluster uses this information to distribute primary shards and their replica shards in different racks or zones. This ensures data availability if a rack or zone fails.

Discovery service

The discovery service provides an IP-based connection management service used when connecting to Redis Enterprise Software databases. It lets your application discover which node hosts the database endpoint. The discovery service API complies with the [Redis Sentinel API](#).

Updated: February 8, 2023

Database clustering

Open source [Redis](#) is a single-threaded process to provide speed and simplicity. A single Redis process is bound by the CPU core that it is running on and available memory on the server.

Redis Enterprise Software supports database clustering to allow customers to spread the load of a Redis process over multiple cores and the RAM of multiple servers. A database cluster is a set of Redis processes where each process manages a subset of the database keyspace.

The keyspace of a Redis Enterprise cluster is partitioned into database shards. Each shard resides on a single node and is managed by that node. Each node in a Redis database cluster can manage multiple shards. The key space in the shards is divided into hash slots. The slot of a key is determined by a hash of the key name or part of the key name.

Database clustering is transparent to the Redis client that connects to the database. The Redis client accesses the database through a single endpoint that automatically routes all operations to the relevant shards. You can connect an application to a single Redis process or a clustered database without any difference in the application logic.

Terminology

In clustering, these terms are commonly used:

- Tag or Hash Tag - A part of the key that is used in the hash calculation.
- Slot or Hash Slot - The result of the hash calculation.
- Shard - Redis process that is part of the Redis clustered database.

When to use clustering (sharding)

Clustering is an efficient way of scaling Redis that should be used when:

- The dataset is large enough to benefit from using the RAM resources of more than one node. When a dataset is more than 25 GB (50 GB for RoF), we recommend that you enable clustering to create multiple shards of the database and spread the data requests across nodes.
- The operations performed against the database are CPU-intensive, resulting in performance degradation. By having multiple CPU cores manage the database's shards, the load of operations is distributed among them.

Number of shards

When enabling database clustering, you can set the number of database shards. The minimum number of shards per database is 2 and the maximum depends on the subscription you purchased.

After you enable database clustering and set the number of shards, you cannot deactivate database clustering or reduce the number of shards. You can only increase the number of shards by a multiple of the current number of shards. For example, if the current number of shards is 3, you can increase the number of shards to 6, 9, or 12.

Supported hashing policies

Standard hashing policy

When using the standard hashing policy, a clustered database behaves just like a standard open source Redis cluster:

- **Keys with a hash tag:** a key's hash tag is any substring between '{' and '}' in the key's name. That means that when a key's name includes the pattern '{...}', the hash tag is used as input for the hashing function. For example, the following key names have the same hash tag and would, therefore, be mapped to the same slot: foo{bar}, {bar}baz, foo{bar}baz.
- **Keys without a hash tag:** when a key does not contain the '{...}' pattern, the entire key's name is used for hashing.

You can use the '{...}' pattern to direct related keys to the same hash slot so that multi-key operations can be executed on these keys. On the other hand, not using a hash tag in the key's name results in a (statistically) even distribution of keys across the keyspace's shards. If your application does not perform multi-key operations, you do not need to construct key names with hash tags.

Custom hashing policy

You can configure a custom hashing policy for a clustered database. A custom hashing policy is required when different keys need to be kept together on the same shard to allow multi-key operations. The custom hashing policy is provided through a set of Perl Compatible Regular Expressions (PCRE) rules that describe the dataset's key name patterns.

To configure a custom hashing policy, enter the regular expression (RegEx) rules that identify the substring in the key's name - hash tag - on which hashing is done. The hash tag is denoted in the RegEx by the use of the `tag` named subpattern. Different keys that have the same hash tag are stored and managed in the same slot.

After you enable the custom hashing policy, the following default RegEx rules are implemented. Update these rules to fit your specific logic:

RegEx Rule	Description
.*{(?:<tag>.*)}.*	Hashing is done on the substring between the curly braces.
(?:<tag>.*)	The entire key's name is used for hashing.

You can modify existing rules, add new ones, delete rules, or change their order to suit your application's requirements.

Custom hashing policy notes and limitations

1. You can define up to 32 RegEx rules, each up to 256 characters.
2. RegEx rules are evaluated in order, and the first rule matched is used. Therefore, you should place common key name patterns at the beginning of the rule list.
3. Key names that do not match any of the RegEx rules trigger an error.
4. The `.*(?:<tag>)` RegEx rule forces keys into a single slot because the hash key is always empty. Therefore, when used, this should be the last, catch-all

rule.

5. The following flag is enabled in the regular expression parser: PCRE_ANCHORED: the pattern is constrained to match only at the start of the string being searched.

Changing the hashing policy

The hashing policy of a clustered database can be changed. However, most hashing policy changes trigger the deletion (FLUSHDB) of the data before they can be applied.

Examples of such changes include:

- Changing the hashing policy from standard to custom or conversely, custom to standard.
- Changing the order of custom hashing policy rules.
- Adding new rules in the custom hashing policy.
- Deleting rules from the custom hashing policy.



Note: The recommended workaround for updates that are not enabled, or require flushing the database, is to back up the database and import the data to a newly configured database.

Multi-key operations

Operations on multiple keys in a clustered database are supported with the following limitations:

- **Multi-key commands:** Redis offers several commands that accept multiple keys as arguments. In a clustered database, most multi-key commands are not allowed across slots. The following multi-key commands are allowed across slots: DEL, MSET, MGET, EXISTS, UNLINK, TOUCH

In Active-Active databases, multi-key write commands (DEL, MSET, UNLINK) can only be run on keys that are in the same slot. However, the following multi-key commands are allowed across slots in Active-Active databases: MGET, EXISTS, and TOUCH.

Commands that affect all keys or keys that match a specified pattern are allowed in a clustered database, for example: FLUSHDB, FLUSHALL, KEYS



Note: When using these commands in a sharded setup, the command is distributed across multiple shards and the responses from all shards are combined into a single response.

- **Geo commands:** For the [GEORADIUS](#) and [GEORADIUSBYMEMBER](#) commands, the STORE and STOREDIST options can only be used when all affected keys reside in the same slot.
- **Transactions:** All operations within a WATCH / MULTI / EXEC block should be performed on keys that are mapped to the same slot.
- **Lua scripts:** All keys used by a Lua script must be mapped to the same slot and must be provided as arguments to the EVAL / EVALSHA commands (as per the Redis specification). Using keys in a Lua script that were not provided as arguments might violate the sharding concept but do not result in the proper violation error being returned.
- **Renaming/Copy keys:** The use of the RENAME / RENAMENX / COPY commands is allowed only when the key's original and new values are mapped to the same slot.

Updated: April 4, 2023

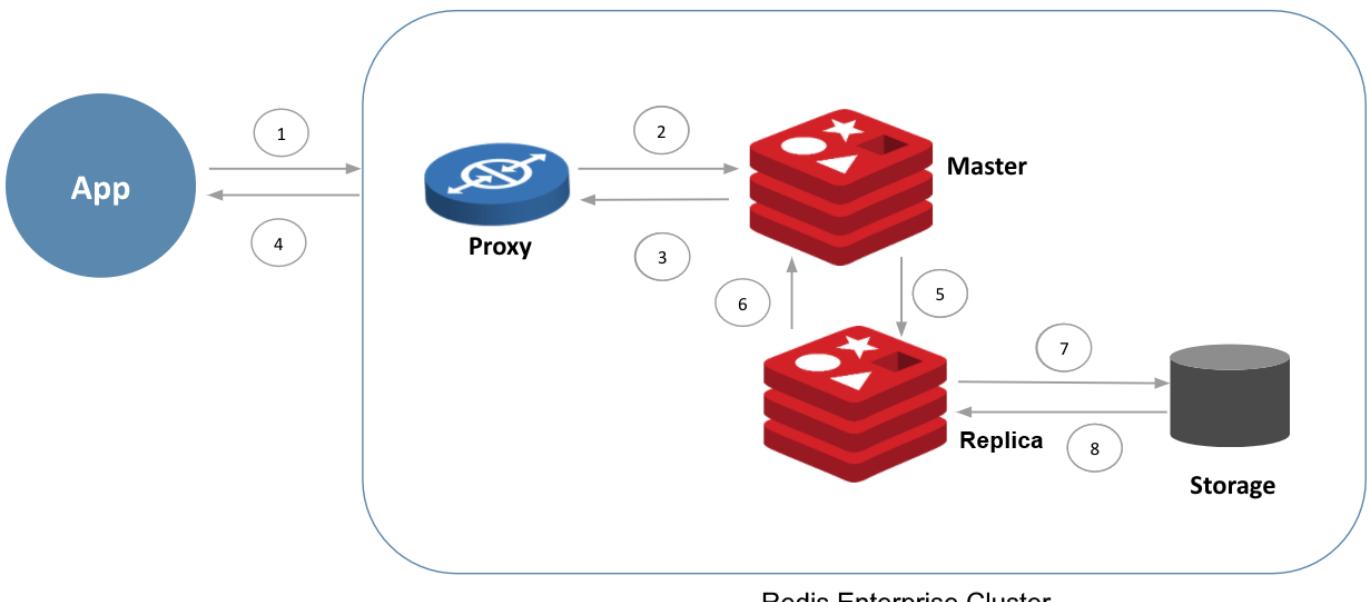
Consistency during replication

Redis Enterprise Software comes with the ability to replicate data to another database instance for high availability and persist in-memory data on disk permanently for durability. With the WAIT command, you can control the consistency and durability guarantees for the replicated and persisted database.

Any updates that are issued to the database are typically performed with the following flow shown below;

1. Application issues a write
2. Proxy communicates with the correct primary (also known as master) "shard" in the system that contains the given key
3. The shard writes the data and sends an acknowledgment to the proxy
4. The proxy sends the acknowledgment back to the application
5. The write is communicated from master to replica
6. Replica acknowledges the write back to the master
7. The write to a replica is persisted to disk
8. The write is acknowledged within the replica

Weak Consistency

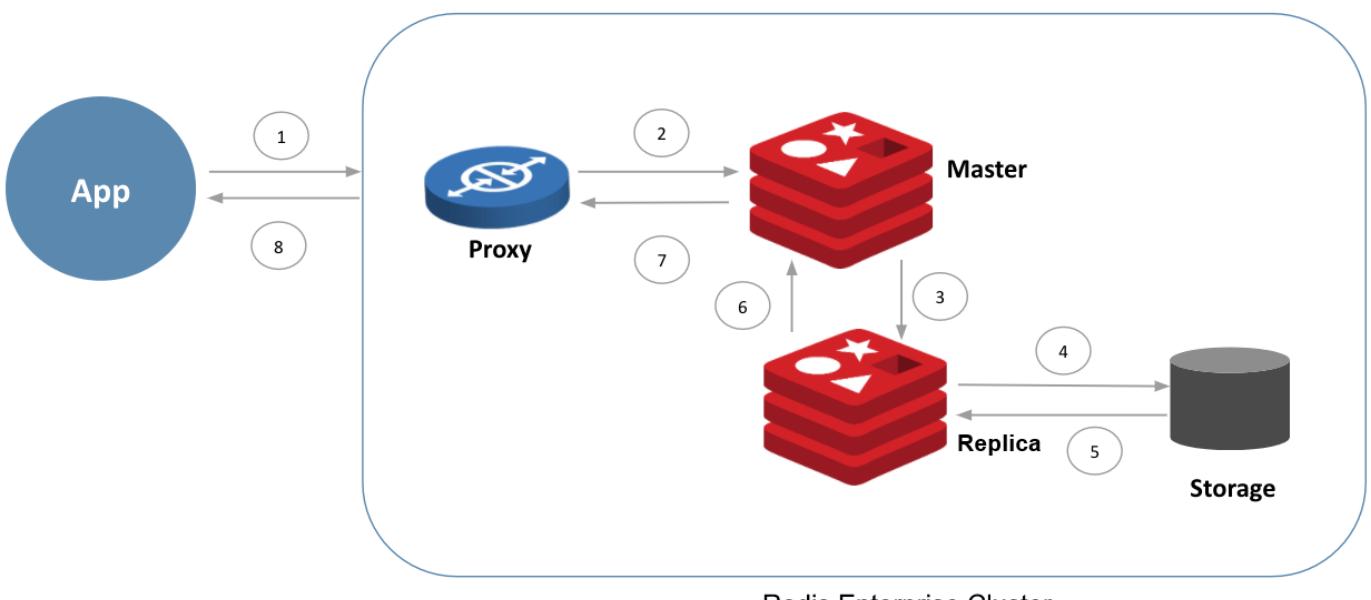


With the WAIT command, applications can ask to wait for acknowledgments only after replication or persistence is confirmed on the replica. The flow of a write operation with the WAIT command is shown below:

1. Application issues a write,
2. Proxy communicates with the correct master "shard" in the system that contains the given key,
3. Replication communicated the update to the replica shard.
4. Replica persists the update to disk (assuming AOF every write setting is selected).
- 5-8. The acknowledgment is sent back from the replica all the way to the proxy with steps 5 to 8.

With this flow, the application only gets the acknowledgment from the write after durability is achieved with replication to the replica and to the persistent storage.

Strong Consistency



With the WAIT command, applications can have a guarantee that even under a node failure or node restart, an acknowledged write is recorded.

See the WAIT command for details on the new durability and consistency options.

Discovery service

The Discovery Service provides an IP-based connection management service used when connecting to Redis Enterprise Software databases. When used in conjunction with Redis Enterprise Software's other high availability features, the Discovery Service assists an application scope with topology changes such as adding, removing of nodes, node failovers and so on. It does this by providing your application with the ability to easily discover which node hosts the database endpoint. The API used for discovery service is compliant with the Redis Sentinel API.

Discovery Service is an alternative for applications that do not want to depend on DNS name resolution for their connectivity. Discovery Service and DNS based connectivity are not mutually exclusive. They can be used side by side in a given cluster where some clients can use Discovery Service based connection while others can use DNS name resolution when connecting to databases.

How discovery service works

The Discovery Service is available for querying on each node of the cluster, listening on port 8001. To employ it, your application utilizes a [Redis Sentinel enabled client library](#) to connect to the Discovery Service and request the endpoint for the given database. The Discovery Service replies with the database's endpoint for that database. In case of a node failure, the Discovery Service is updated by the cluster manager with the new endpoint and clients unable to connect to the database endpoint due to the failover, can re-query the discovery service for the new endpoint for the database.

The Discovery Service can return either the internal or external endpoint for a database. If you query the discovery service for the endpoint of a database named "db1", the Discovery Service returns the external endpoint information by default. If only an internal endpoint exists with no external endpoint the default behavior is to return the internal endpoint. The "@internal" is added to the end of the database name to explicitly ask for the internal endpoint. To query the internal endpoint explicitly with database name "db1", you can pass in the database name as "db1@internal".

If you'd like to examine the metadata returned from Redis Enterprise Software Discovery Service you can connect to port 8001 with redis-cli utility and execute "SENTINEL masters". Following is a sample output from one of the nodes of a Redis Enterprise Software cluster:

```
$ ./redis-cli -p 8001
127.0.0.1:8001> SENTINEL masters
1) 1) "name"
2) "db1@internal"
3) "ip"
4) "10.0.0.45"
5) "port"
6) "12000"
7) "flags"
8) "master,disconnected"
9) "num-other-sentinels"
10) "0"
2) 1) "name"
2) "db1"
3) "ip"
4) "10.0.0.45"
5) "port"
6) "12000"
7) "flags"
8) "master,disconnected"
9) "num-other-sentinels"
10) "0"
```

It is important to note that, the Discovery Service is not a full implementation of the [Redis Sentinel protocol](#). There are aspects of the protocol that are not applicable or would be duplication with existing technology in Redis Enterprise Software. The Discovery Service implements only the parts required to provide applications with easy High Availability, be compatible with the protocol, and not rely on DNS to derive which node in the cluster to communicate with.



Note: To use Redis Sentinel, every database name must be unique across the cluster.

Redis client support

We recommend these clients that are tested for use with the [Discovery Service](#) that uses the Redis Sentinel API:

- [Redis-py](#) (Python redis client)
- [HiRedis](#) (C redis client)
- [Jedis](#) (Java redis client)
- [Ioredis](#) (NodeJS redis client)

If you need to use another client, consider using [Sentinel Tunnel](#) to discover the current Redis master with Sentinel and create a TCP tunnel between a local port on the client and the master.



Note: Redis Sentinel API can return endpoints for both master and replica endpoints. Discovery Service only supports master endpoints and does not support returning replica endpoints for a database.

Updated: August 31, 2022

Database replication

Database replication helps ensure high availability. When replication is enabled, your dataset is replicated to a replica shard, which is constantly synchronized with the primary shard. If the primary shard fails, an automatic failover happens and the replica shard is promoted. That is, it becomes the new primary shard.

When the old primary shard recovers, it becomes the replica shard of the new primary shard. This auto-failover mechanism guarantees that data is served with minimal interruption.

You can tune your high availability configuration with:

- [Rack/Zone Awareness](#) - When rack-zone awareness is used additional logic ensures that master and replica shards never share the same rack, thus ensuring availability even under loss of an entire rack.
- [High Availability for Replica Shards](#) - When high availability for replica shards is used, the replica shard is automatically migrated on node failover to maintain high availability.



Warning - Enabling replication has implications for the total database size, as explained in [Database memory limits](#).

Auto Tiering replication considerations

We recommend that you set the sequential replication feature using `radmin`. This is due to the potential for relatively slow replication times that can occur with Auto Tiering enabled databases. In some cases, if sequential replication is not set up, you may run out of memory.

While it does not cause data loss on the primary shards, the replication to replica shards may not succeed as long as there is high write-rate traffic on the primary and multiple replications at the same time.

The following `radmin` command sets the number of primary shards eligible to be replicated from the same cluster node, as well as the number of replica shards on the same cluster node that can run the replication process at any given time.

The recommended sequential replication configuration is two, i.e.:

```
radmin tune cluster max_redis_forks 1 max_slave_full_syncs 1
```



Note: This means that at any given time, only one primary and one replica can be part of a full sync replication process.

Database replication backlog

Redis databases that use [replication for high availability](#) maintain a replication backlog (per shard) to synchronize the primary and replica shards of a database. By default, the replication backlog is set to one percent (1%) of the database size divided by the database number of shards and ranges between 1MB to 250MB per shard. Use the `radmin` and the `crdb-cli` utilities to control the size of the replication backlog. You can set it to `auto` or set a specific size.

The syntax varies between regular and Active-Active databases.

For a regular Redis database:

```
radmin tune db <db:id | name> repl_backlog <Backlog size in MB | 'auto'>
```

For an Active-Active database:

```
crdb-cli crdb update --crdb-guid <crdb_guid> --default-db-config "{\"repl_backlog_size\": <size in MB | 'auto'>}"
```

Active-Active replication backlog

In addition to the database replication backlog, Active-Active databases maintain a backlog (per shard) to synchronize the database instances between clusters. By default, the Active-Active replication backlog is set to one percent (1%) of the database size divided by the database number of shards, and ranges between 1MB to 250MB per shard. Use the [crdb-cli](#) utility to control the size of the CRDT replication backlog. You can set it to `auto` or set a specific size:

```
crdb-cli crdb update --crdb-guid <crdb_guid> --default-db-config "{\"crdt_repl_backlog_size\": <size in MB | 'auto'>}"
```

For Redis Software versions earlier than 6.0.20: The replication backlog and the CRDT replication backlog defaults are set to 1MB and cannot be set dynamically with ‘auto’ mode. To control the size of the replication log, use [rladmin](#) to tune the local database instance in each cluster.

```
rladmin tune db <db:id | name> repl_backlog <Backlog size in MB (or if ending with bytes, KB or GB, in the respective unit)>
```

Updated: August 17, 2023

Memory and performance

Redis Enterprise Software has multiple mechanisms in its architecture to help optimize storage and performance.

Memory limits

Database memory limits define the maximum size your database can reach across all database replicas and [shards](#) on the cluster. Your memory limit will also determine the number of shards you’ll need.

Besides your dataset, the memory limit must also account for replication, Active-Active overhead, and module overhead, and a number of other factors. These can significantly increase your database size, sometimes increasing it by four times or more.

For more information on memory limits, see [Database memory limits](#).

Eviction policies

When a database exceeds its memory limit, eviction policies determine which data is removed. The eviction policy removes keys based on frequency of use, how recently used, randomly, expiration date, or a combination of these factors. The policy can also be set to `noeviction` to return a memory limit error when trying to insert more data.

The default eviction policy for databases is `volatile-lru` which evicts the least recently used keys out of all keys with the `expire` field set. The default for Active-Active databases is `noeviction`.

For more information, see [eviction policies](#).

Database persistence

Both RAM memory and flash memory are at risk of data loss if a server or process fails. Persisting your data to disk helps protect it against loss in those situations. You can configure persistence at the time of database creation, or by editing the database’s configuration.

There are two main types of persistence strategies in Redis Enterprise Software: append-only files (AoF) and snapshots.

Append-only files (AoF) keep a record of data changes and writes each change to the end of a file, allowing you to recover the dataset by replaying the writes in the append-only log.

Snapshots capture all the data as it exists in one moment in time and writes it to disk, allowing you to recover the entire dataset as it existed at that moment in time.

For more info on data persistence see [Database persistence with Redis Enterprise Software](#) or [Durable Redis](#).

Auto Tiering

By default, Redis Enterprise Software stores your data entirely in [RAM](#) for improved performance. The [Auto Tiering](#) feature enables your data to span both RAM and [SSD](#) storage ([flash memory](#)). Keys are always stored in RAM, but Auto Tiering manages the location of their values. Frequently used (hot) values are stored on RAM, but infrequently used (warm) values are moved to flash memory. This saves on expensive RAM space, which give you comparable

performance at a lower cost for large datasets.

For more info, see [Auto Tiering](#).

Shard placement

The location of the primary and replica shards on the cluster nodes can impact your database performance. Primary shards and their corresponding replica shards are always placed on separate nodes for data resiliency and high availability. The shard placement policy helps to maintain optimal performance and resiliency.

Redis Enterprise Software has two shard placement policies available:

- **dense**: puts as many shards as possible on the smallest number of nodes
- **sparse**: spread the shards across as many nodes as possible

For more info about the shard placement policy, see [Shard placement policy](#)

Metrics

From the Redis Enterprise Software admin console, you can monitor the performance of your clusters, nodes, databases, and shards with real-time metrics. You can also enable alerts for node, cluster, or database events such as high memory usage or throughput.

With the Redis Enterprise Software API, you can also integrate Redis Enterprise metrics into other monitoring environments, such as Prometheus.

For more info about monitoring with Redis Enterprise Software, see [Monitoring with metrics and alerts](#), and [Memory statistics](#).

Scaling databases

Each Redis Enterprise cluster can contain multiple databases. In Redis, databases represent data that belong to a single application, tenant, or microservice. Redis Enterprise is built to scale to 100s of databases per cluster to provide flexible and efficient multi-tenancy models.

Each database can contain few or many Redis shards. Sharding is transparent to Redis applications. Master shards in the database process data operations for a given subset of keys. The number of shards per database is configurable and depend on the throughput needs of the applications. Databases in Redis Enterprise can be resharded into more Redis shards to scale throughput while maintaining sub-millisecond latencies. Resharding is performed without downtime.

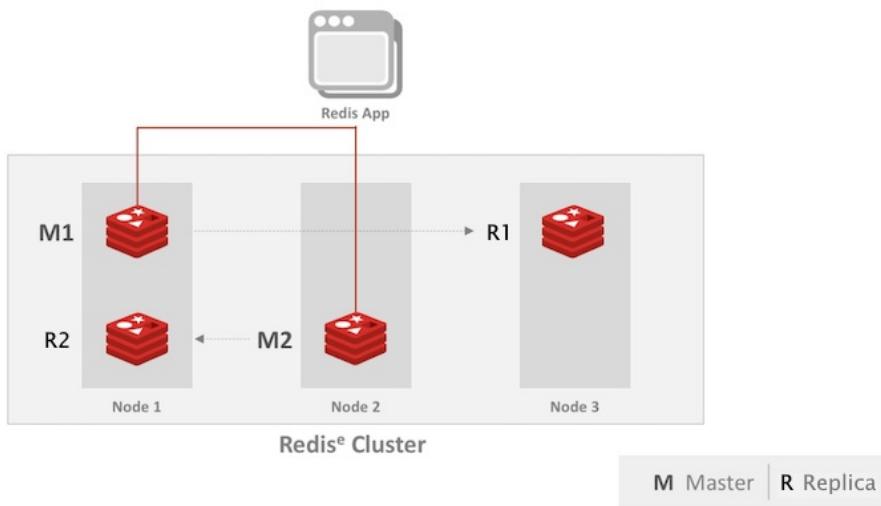


Figure 2 Redis Enterprise places master node (M) and replicas (R) in separate nodes, racks and zones and use in-memory replication to protect data against failures.

In Redis Enterprise, each database has a quota of RAM. The quota cannot exceed the limits of the RAM available on the node. However, with Redis Enterprise Flash, RAM is extended to the local flash drive (SATA, NVMe SSDs etc). The total quota of the database can take advantage of both RAM and Flash drive. The administrator can choose the RAM vs Flash ratio and adjust that anytime in the lifetime of the database without downtime.

With Auto Tiering, instead of storing all keys and data for a given shard in RAM, less frequently accessed values are pushed to flash. If applications need to access a value that is in flash, Redis Enterprise automatically brings the value into RAM. Depending on the flash hardware in use, applications experience slightly higher latency when bringing values back into RAM from flash. However subsequent accesses to the same value is fast, once the value is in RAM.

Eviction policy

The eviction policy determines what happens when a database reaches its memory limit.

To make room for new data, older data is *evicted* (removed) according to the selected policy.

To prevent this from happening, make sure your database is large enough to hold all desired keys.

Eviction Policy	Description
noeviction	New values aren't saved when memory limit is reached
allkeys-lru	When a database uses replication, this applies to the primary database
allkeys-lfu	Keeps most recently used keys; removes least recently used (LRU) keys
allkeys-random	Keeps frequently used keys; removes least frequently used (LFU) keys
volatile-lru	Randomly removes keys
volatile-lfu	Removes least recently used keys with <code>expire</code> field set to true
volatile-random	Removes least frequently used keys with <code>expire</code> field set to true
volatile-ttl	Randomly removes keys with <code>expire</code> field set to true
volatile-ttl	Removes least frequently used keys with <code>expire</code> field set to true and the shortest remaining time-to-live (TTL) value

Eviction policy defaults

`volatile-lru` is the default eviction policy for most databases.

The default policy for [Active-Active databases](#) is `noeviction` policy.

Active-Active database eviction

The eviction policy mechanism for Active-Active databases kicks in earlier than for standalone databases because it requires propagation to all participating clusters. The eviction policy starts to evict keys when one of the Active-Active instances reaches 80% of its memory limit. If memory usage continues to rise while the keys are being evicted, the rate of eviction will increase to prevent reaching the Out-of-Memory state. As with standalone Redis Enterprise databases, Active-Active eviction is calculated per shard. To prevent over eviction, internal heuristics might prevent keys from being evicted when the shard reaches the 80% memory limit. In such cases, keys will get evicted only when shard memory reaches 100%.

In case of network issues between Active-Active instances, memory can be freed only when all instances are in sync. If there is no communication between participating clusters, it can result in eviction of all keys and the instance reaching an Out-of-Memory state.



Note: Data eviction policies are not supported for Active-Active databases with Auto Tiering .

Avoid data eviction

To avoid data eviction, make sure your database is large enough to hold required values.

For larger databases, consider using [Auto Tiering](#) .

Auto Tiering stores actively-used data (also known as *hot data*) in RAM and the remaining data in flash memory (SSD). This lets you retain more data while ensuring the fastest access to the most critical data.

Updated: August 17, 2023

Database memory limits

When you set a database's memory limit, you define the maximum size the database can reach in the cluster, across all database replicas and shards, including both primary and replica shards.

If the total size of the database in the cluster reaches the memory limit, the data eviction policy is applied.

Factors for sizing

Factors to consider when sizing your database:

- **dataset size**: you want your limit to be above your dataset size to leave room for overhead.
- **database throughput**: high throughput needs more shards, leading to a higher memory limit.
- **modules**: using modules with your database consumes more memory.
- **database clustering**: enables you to spread your data into shards across multiple nodes.
- **database replication**: enabling replication doubles memory consumption.

Additional factors for Active-Active databases:

- **Active-Active replication**: enabling Active-Active replication requires double the memory of regular replication, which can be up to two times (2x) the original data size per instance.
- **database replication backlog** for synchronization between shards. By default, this is set to 1% of the database size.
- **Active-Active replication backlog** for synchronization between clusters. By default, this is set to 1% of the database size.

It's also important to know Active-Active databases have a lower threshold for activating the eviction policy, because it requires propagation to all participating clusters. The eviction policy starts to evict keys when one of the Active-Active instances reaches 80% of its memory limit.

Additional factors for databases with Auto Tiering enabled:

- **database persistence**: Auto Tiering uses dual database persistence where both the primary and replica shards persist to disk. This may add some processor and network overhead, especially in cloud configurations with network attached storage.

What happens when Redis Enterprise Software is low on RAM?

Redis Enterprise Software manages node memory so that data is entirely in RAM (unless using Auto Tiering). If not enough RAM is available, Redis Enterprise prevents adding more data into the databases.

Redis Enterprise Software protects the existing data and prevents the database from being able to store data into the shards.

You can configure the cluster to move the data to another node, or even discard it according to the [eviction policy](#) set on each database by the administrator.

[Auto Tiering](#) manages memory so that you can also use flash memory (SSD) to store data.

Order of events for low RAM

1. If there are other nodes available, your shards migrate to other nodes.
2. If the eviction policy allows eviction, shards start to release memory, which can result in data loss.
3. If the eviction policy does not allow eviction, you'll receive out of memory (OOM) messages.
4. If shards can't free memory, Redis Enterprise relies on the OS processes to stop replicas, but tries to avoid stopping primary shards.

We recommend that you have a [monitoring platform](#) that alerts you before a system gets low on RAM. You must maintain sufficient free memory to make sure that you have a healthy Redis Enterprise installation.

Memory metrics

The admin console provides metrics that can help you evaluate your memory use.

- Free RAM
- RAM fragmentation
- Used memory
- Memory usage
- Memory limit

See [console metrics](#) for more detailed information.

Related info

- [Memory and performance](#)
- [Disk sizing for heavy write scenarios](#)
- [Turn off services to free system memory](#)
- [Eviction policy](#)
- [Shard placement policy](#)
- [Database persistence](#)

Shard placement policy

In Redis Enterprise Software, the location of master and replica shards on the cluster nodes can impact the database and node performance. Master shards and their corresponding replica shards are always placed on separate nodes for data resiliency. The shard placement policy helps to maintain optimal performance and resiliency.

In addition to the shard placement policy, considerations that determine shard placement are:

- Separation of master and replica shards
- Available persistence and Auto Tiering storage
- [Rack awareness](#)
- Memory available to host the database when fully populated

The shard placement policies are:

- **dense** - Place as many shards as possible on the smallest number of nodes to reduce the latency between the proxy and the database shards; Recommended for Redis on RAM databases to optimize memory resources
- **sparse** - Spread the shards across as many nodes in the cluster as possible to spread the traffic across cluster nodes; Recommended for databases with Auto Tiering enabled to optimize disk resources

When you create a Redis Enterprise Software cluster, the default shard placement policy (dense) is assigned to all databases that you create on the cluster.

You can:

- Change the default shard placement policy for the cluster to **sparse** so that the cluster applies that policy to all databases that you create
- Change the shard placement policy for each database after the database is created

Shard placement policies

Dense shard placement policy

In the dense policy, the cluster places the database shards on as few nodes as possible. When the node is not able to host all of the shards, some shards are moved to another node to maintain optimal node health.

For example, for a database with two master and two replica shards on a cluster with three nodes and a dense shard placement policy, the two master shards are hosted on one node and the two replica shards are hosted on another node.

For Redis on RAM databases without the OSS cluster API enabled, use the dense policy to optimize performance.

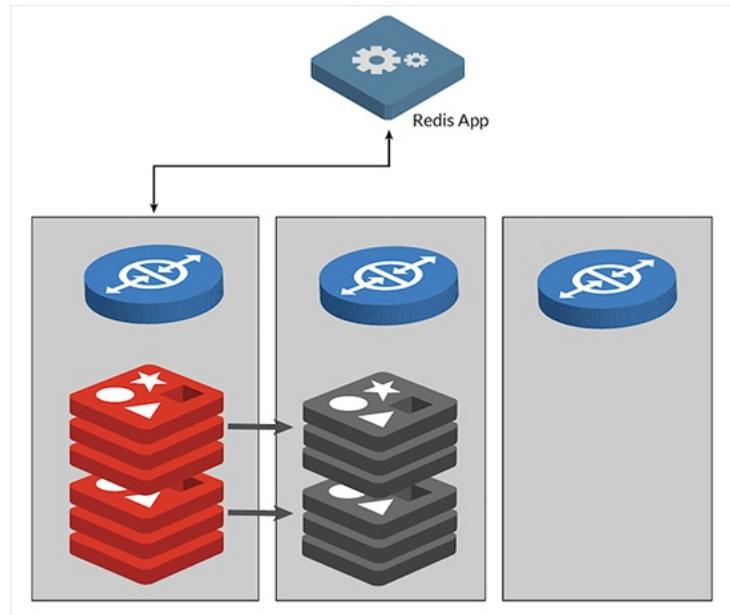


Figure: Three nodes with two master shards (red) and two replica shards (grey) with a dense placement policy

Sparse shard placement policy

In the sparse policy, the cluster places shards on as many nodes as possible to distribute the shards of a database across all available nodes. When all nodes have database shards, the shards are distributed evenly across the nodes to maintain optimal node health.

For example, for a database with two master and two replica shards on a cluster with three nodes and a sparse shard placement policy:

- Node 1 hosts one of the master shards
- Node 2 hosts the replica for the first master shard
- Node 3 hosts the second master shard
- Node 1 hosts for the replica shard for master shard 2

For Redis on RAM databases with OSS cluster API enabled and for databases with Auto Tiering enabled, use the sparse policy to optimize performance.

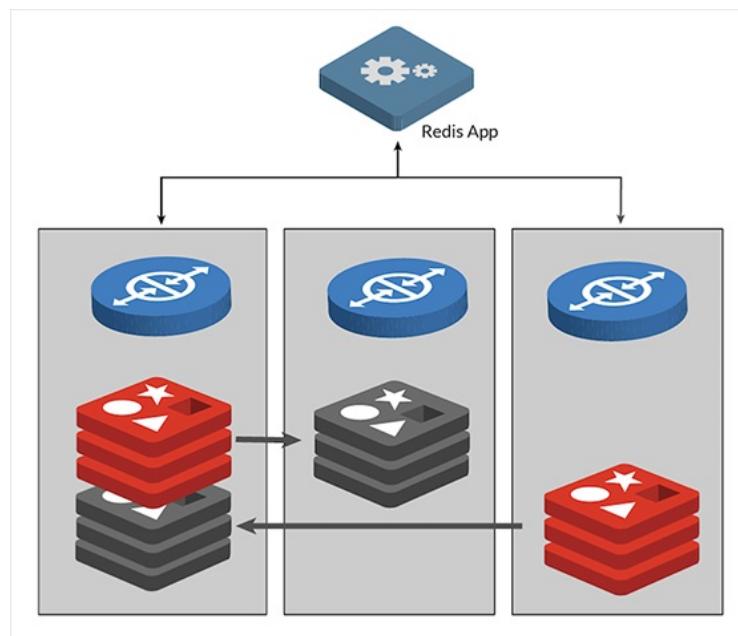


Figure: Three nodes with two master shards (red) and two replica shards (grey) with a sparse placement policy

Related articles

You can [configure the shard placement policy](#) for each database.

Updated: August 17, 2023

Manage clusters

You can manage your Redis Enterprise Software clusters with several different tools:

- Admin console (the web-based user interface)
- Command-line tools ([rladmin](#), [redis-cli](#), [crdb-cli](#))
- [REST API](#)

Manage your cluster

[Set up cluster](#)

Set up a new cluster using the admin console.

[Add a node](#)

Add a node to an existing Redis Enterprise cluster.

[Configure clusters](#)

Change [cluster settings](#).

[Optimize clusters](#)

Find information and configuration settings to improve the performance of Redis Enterprise Software.

Monitor cluster

Monitor the cluster and database activity with [cluster logs](#) and metrics.

Updated: December 12, 2022

Set up a new cluster

A Redis Enterprise Software cluster typically consists of several nodes. For production deployments, we recommend an uneven number of nodes, with a minimum of three.

 **Note:** In a cluster that consists of only one node, some features and capabilities are not enabled, such as database replication that provides high availability.

To set up a new cluster, you must first [install the Redis Enterprise Software package](#) and then set up the cluster as described below. After the cluster is created you can [add multiple nodes to the cluster](#).

To create a cluster:

1. In a browser, navigate to `https://<name or IP address of the machine with Redis Enterprise Software installed>:8443/new`. For example, if you installed Redis Enterprise Software on a machine with IP address 10.0.1.34, then navigate to <https://10.0.1.34:8443/new>.

 **Note:**

- The management UI uses a [self-signed certificate for TLS encryption](#).
- If the machine has both an internal IP address and an external IP address, use the external IP address to access the setup UI.

2. Select **Create new cluster**.
3. Enter an email and password for the administrator account, then select **Next** to proceed to cluster setup.
4. Enter your cluster license key if you have one. Otherwise, the cluster uses the trial license by default.

5. In the **Configuration** section:

1. For **FQDN (Fully Qualified Domain Name)**, enter a unique name for the cluster.

See the [instructions for DNS setup](#) to make sure your cluster is reachable by name.

2. Choose whether to [Enable private & public endpoints support](#).
3. Choose whether to [Enable rack-zone awareness](#).

6. Click **Next**.

7. Configure storage and network settings:

1. Enter a path for [Ephemeral storage](#), or leave the default path.
2. Enter a path for [Persistent storage](#), or leave the default path.
3. To enable [Auto Tiering](#), select [Enable flash storage](#) and enter the path to the flash storage.
4. If the cluster is configured to support [rack-zone awareness](#), set the **Rack-zone ID** for the new node.
5. If your machine has multiple IP addresses, assign a single IPv4 type address for **Node-to-node communication (internal traffic)** and multiple IPv4/IPv6 type addresses for **External traffic**.

8. Select **Create cluster**.

9. Click **OK** to confirm that you are aware of the replacement of the HTTPS TLS certificate on the node, and proceed through the browser warning.

After a short wait, your cluster is created and you can sign in to the admin console.

You can now access any of the management capabilities, including:

- [Creating a new database](#)
- [Joining a new node to a cluster](#)

Add a cluster node

When you install Redis Enterprise Software on the first node of a cluster, you create the new cluster. After you install the first node, you can add more nodes to the cluster.



Note:

Before you add a node to the cluster:

- The clocks on all nodes must always be [synchronized](#).
If the clock in the node you are trying to join to the cluster is not synchronized with the nodes already in the cluster, the action fails and an error message is shown indicating that you must synchronize the clocks first.
- You must [update the DNS records](#) each time a node is added or replaced.
- We recommend that you add nodes one after the other rather than in parallel to avoid errors that occur because the connection to the other nodes in the cluster cannot be verified.

To add a node to an existing cluster:

1. [Install the Redis Enterprise Software installation package](#) on a clean installation of a [supported operating system](#).
2. To connect to the management UI of the new Redis Enterprise Software installation, go to: <<https://URL or IP address:8443/new>>

For example, if you installed Redis Enterprise Software on a machine with IP address 10.0.1.34, go to <https://10.0.1.34:8443/new>.



Tip - The management UI uses TLS encryption with a default certificate. You can also [replace the TLS certificate](#) with a custom certificate.

3. Select **Join cluster**.
4. For **Cluster identification**, enter the internal IP address or DNS name of a node that is a cluster member.

If the node only has one IP address, enter that IP address.

5. For **Cluster sign in**, enter the credentials of the cluster administrator.

The cluster administrator is the user account that you create when you configure the first node in the cluster.

6. Click **Next**.

7. Configure storage and network settings:

1. Enter a path for [Ephemeral storage](#), or leave the default path.
2. Enter a path for [Persistent storage](#), or leave the default path.
3. To enable [Auto Tiering](#), select **Enable flash storage** and enter the path to the flash storage.
4. If the cluster is configured to support [rack-zone awareness](#), set the **Rack-zone ID** for the new node.
5. If your machine has multiple IP addresses, assign a single IPv4 type address for **Node-to-node communication (internal traffic)** and multiple IPv4/IPv6 type addresses for **External traffic**.

8. Select **Join cluster**.

The node is added to the cluster. You can see it in the list of nodes in the cluster.

If you see an error when you add the node, try adding the node again.



Tip - We recommend that you run the [rlcheck utility](#) to verify that the node is functioning properly.

Configure clusters

You can manage your Redis Enterprise Software clusters with several different tools:

- Admin console (the web-based user interface)
- Command-line tools ([rladmin](#), [redis-cli](#), [crdb-cli](#))
- REST API

Cluster settings

You can view and set various cluster settings such as cluster name, email service, time zone, and license.

Cluster license keys

The cluster key (or license) enables features and capacity within Redis Enterprise Software

Synchronize cluster node clocks

Sync node clocks to avoid problems with internal cluster communication.

Rack-zone awareness in Redis Enterprise Software

Rack-zone awareness ensures high-availability in the event of a rack or zone failure.

Updated: July 7, 2022

Cluster settings

You can view and set various cluster settings, such as cluster name, email service, time zone, and license, on the [Cluster > Configuration](#) page.

General configuration tab

Upload cluster license key

After purchasing a cluster license and if your account has the “Admin” role, you can upload the cluster license key, either during initial cluster creation or at any time afterward. The license key defines various cluster settings, such as the maximum number of shards you can have in the cluster. For more detailed information see [Cluster license keys](#).

View max number of allowed shards

The maximum number of allowed shards, which is determined by the cluster license key, appears in the **Max number of shards** field in the **License** section.

View cluster name

The cluster name appears in the **Cluster name** field in the **License** section. This gives a common name that your team or Redis support can refer to. It is especially helpful if you have multiple clusters.

Set time zone

You can set your time zone in the **Time zone** field. This is recommended to make sure the date, time fields, and log entries are shown in your preferred time zone.

Alert settings tab

The **Alert Settings** tab lets you configure alerts that are relevant to the entire cluster, such as alerts for cluster utilization, nodes, node utilization, security, and database utilization.

You can also configure email server settings and [send alerts by email](#) to relevant users.

Configure email server settings

To enable email alerts:

1. Enter your email server details in the **Email server settings** section.
2. Select a connection security method:
 - TLS/SSL
 - STARTTLS
 - None
3. Send a test email to verify your email server settings.

Updated: August 17, 2023

Cluster license keys

The cluster key (or license) enables features and capacity within Redis Enterprise Software. You can add or update a cluster key at any time in a cluster lifecycle. When the cluster does not have a cluster key, the cluster is in trial mode.

Trial mode

Trial mode is limited to thirty days and a total of four shards, including master and replica shards. Any new installation starts its thirty-day clock from the day the cluster setup was done (with the first cluster node provisioned). This mode allows all features to be enabled, including Auto Tiering, during the trial period.

View cluster license key

To view the cluster license key, use:

- Cluster Manager UI
 1. Go to **Cluster > Configuration > General > License** to see the cluster license details.
 2. Select **Change** to view the cluster license key.
- REST API - [GET /v1/license](#)

For a list of returned fields, see the [response section](#).



Note: As of version 7.2, Redis Enterprise will enforce the shards limits by their types (RAM, flash) rather than total number of shards. The flash shards limit only appears in the UI if Auto Tiering is enabled.

Update cluster license



Note: After you add a cluster key, you cannot remove the key to return the cluster to trial mode.

You can update the cluster license key:

- During cluster setup using the admin console or CLI
- After cluster setup using the admin console:
 1. Go to **Cluster > Configuration > General > License**
 2. Select **Change**.
 3. Upload or enter your cluster license key.
 4. Select **Save**.

You can update an existing cluster key at any time. Redis Enterprise checks the validity of it in terms of:

- cluster name
- activation and expiration dates
- shard usage and limits
- features

If saving a new cluster key fails, the operation returns an error including failure reason. In this case, the existing key stays in effect.

Expired cluster license

When the license is expired:

- You cannot do these actions:
 - Change database settings including security and configuration options.
 - Add or remove a database.
 - Upgrade a database to a new version.
 - Add or remove a node.
- You can do these actions:
 - Sign in to the admin console and view settings and metrics at all resolutions for the cluster, nodes, and databases.
 - Change cluster settings including license key, security for administrators, and cluster alerts.
 - Failover when a node fails and explicitly migrate shards between nodes.
 - Upgrade a node to a new version of Redis Enterprise Software.

Monitor cluster license

As of version 7.2, Redis Enterprise exposes the license quotas and the shards consumption metrics in the Cluster Manager UI or via the [Prometheus integration](#).

The ‘cluster_shards_limit’ metric displays the total shard limit by the license by shard type (ram / flash). Examples:

- cluster_shards_limit{cluster="mycluster.local",shard_type="ram"} 100.0
- cluster_shards_limit{cluster="mycluster.local",shard_type="flash"} 50.0

The ‘bdb_shards_used’ metric displays the used shard count by database and by shard type (ram / flash). Examples:

- bdb_shards_used{bdb="2",cluster="mycluster.local",shard_type="ram"} 86.0
- bdb_shards_used{bdb="3",cluster="mycluster.local",shard_type="flash"} 23.0

Updated: August 17, 2023

Synchronize cluster node clocks

To avoid problems with internal cluster communications that can impact your data integrity, make sure that the clocks on all of the cluster nodes are synchronized using Chrony and/or NTP.

When you install Redis Enterprise Software, the install script checks if Chrony or NTP is running. If they are not, the installation process asks for permission to configure a scheduled Cron job. This should make sure that the node's clock is always synchronized. If you did not confirm configuring this job during the installation process, you must use the Network Time Protocol (NTP) regularly to make sure that all server clocks are synchronized.

To synchronize the server clock, run the command that is appropriate for your operating system. For example, in Ubuntu, the following command can be used to synchronize a server's clock to an NTP server:

```
sudo /etc/network/if-up.d/ntpdate
```

If you are using Active-Active databases, you must use [Network Time Service \(ntpd\)](#) to synchronize OS clocks consistent across clusters to handle conflict resolution according to the OS time.

Updated: June 6, 2022

Rack-zone awareness in Redis Enterprise Software

Rack-zone awareness is a Redis Enterprise feature that helps to ensure high-availability in the event of a rack or zone failure.

When you enable rack-zone awareness in a Redis Enterprise Software cluster, you assign a rack-zone ID to each node. This ID is used to map the node to a physical rack or logical zone. The cluster can then ensure that master shards, corresponding replica shards, and associated endpoints are placed on nodes in different racks/zones.

In the event of a rack or zone failure, the replicas and endpoints in the remaining racks/zones are promoted. This ensures high availability when a rack or zone fails.

There is no limitation on the number of rack-zones per cluster; each node can belong to a different rack, or multiple nodes can belong to the same rack.

Rack-zone awareness affects various cluster, node and database-related actions, such as node rebalancing, node removal, node replacement, shard and endpoint migration, and database failover.

Cluster and node configuration

To enable rack-zone awareness, you need to configure it at the cluster, node, and database levels.

First, enable rack-zone awareness when you initially create the cluster.

Now, every time you add a new node to the cluster, define a **rack-zone ID** for the node.

The rack-zone ID must comply with the following rules:

- Maximum length of 63 characters.
- Characters consist of letters, digits, and hyphens ('-'). Underscores ('_') are also accepted as of Redis Enterprise Software [6.4.2-61](#).
- ID starts with a letter and ends with a letter or a digit.

 | Note: Rack-zone IDs are **case-insensitive** (uppercase and lowercase letter are treated as the same).

Node layout

Recall that the recommended minimum number of nodes in a RS deployment is three. For high availability, these three nodes must be distributed across three *distinct* racks or zones.

When using availability zones, note that all three zones should exist within the same region to avoid potential latency issues.

Keep in mind that one of the nodes in your cluster can be a quorum-only node, assuming compute resources are limited. What this means is that the minimum rack-zone aware RS deployment will consist of two data nodes and one quorum-only node, where each of these nodes is situated in a distinct rack or zone.

Database configuration

Once the cluster has been configured to support rack-zone awareness, you can create a rack-zone aware database.

Rack-zone awareness is relevant only for databases that have replication enabled (i.e., databases with replica shards). Once you enable replication for a database, you may also enable rack-zone awareness.

Shard placement without rack-zone awareness

Note that even in the case of a database with rack-zone awareness disabled, the cluster will still ensure that master and replica shards are placed on distinct nodes.

Updated: April 10, 2023

Optimize clusters

Benchmarking Redis Enterprise

Use the `memtier_benchmark` tool to perform a performance benchmark of Redis Enterprise Software.

Disk sizing for heavy write scenarios

Determine the required persistent disk space for heavy throughput databases.

Cluster environment optimization

Optimize your cluster environments for better performance.

Turn off services to free system memory

Turn off services to free memory and improve performance.

Redis OSS Cluster API

Use the Redis OSS Cluster API to improve performance and keep applications current with cluster topology changes.

WAIT command

Use the WAIT command to control the consistency and durability guarantees for the replicated and persisted database.

Updated: February 17, 2023

Benchmarking Redis Enterprise

Use the memtier_benchmark tool to perform a performance benchmark of Redis Enterprise Software.

Prerequisites:

- Redis Enterprise Software installed
- A cluster configured
- A database created

For help with the prerequisites, see the [Redis Enterprise Software quickstart](#).

It is recommended to run memtier_benchmark on a separate node that is not part of the cluster being tested. If you run it on a node of the cluster, be mindful that it affects the performance of both the cluster and memtier_benchmark.

```
/opt/redislabs/bin/memtier_benchmark -s $DB_HOST -p $DB_PORT -a $DB_PASSWORD -t 4 -R --ratio=1:1
```

This command instructs memtier_benchmark to connect to your Redis Enterprise database and generates a load doing the following:

- A 50/50 Set to Get ratio
- Each object has random data in the value

Populate a database with testing data

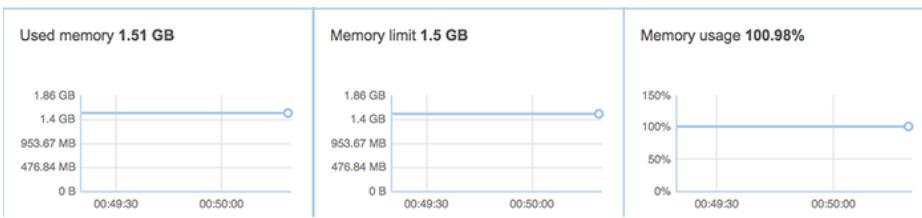
If you need to populate a database with some test data for a proof of concept, or failover testing, etc. here is an example for you.

```
/opt/redislabs/bin/memtier_benchmark -s $DB_HOST -p $DB_PORT -a $DB_PASSWORD -R -n allkeys -d 500 --key-pattern=P:P --ratio=1:0
```

This command instructs memtier_benchmark to connect to your Redis Enterprise database and generates a load doing the following:

- Write objects only, no reads
- A 500 byte object
- Each object has random data in the value
- Each key has a random pattern, then a colon, followed by a random pattern.

Run this command until it fills up your database to where you want it for testing. The easiest way to check is on the database metrics page.



Another use for memtier_benchmark is to populate a database with data for failure testing.

Updated: August 17, 2023

Disk sizing for heavy write scenarios

In extreme write scenarios, when AOF is enabled, the AOF rewrite process may require considerably more disk space for database persistence.

To estimate the required persistent disk space in such cases, use the formula described below.

The required persistent disk space for AOF rewrite purposes in extreme write scenarios, assuming identical shard sizes:

$X(1 + 3Y + Y^2)$ where: X = each shard size Y = number of shards

Following are examples of database configurations and the persistence disk space they would require in this scenario:

	Example 1	Example 2	Example 3	Example 4
Database size (GB)	10	10	40	40
Number of shards	4	16	5	15
Shard size (GB)	2.5	0.625	8	2.67
Required disk space (GB)	73	191	328	723

For disk size requirements in standard usage scenarios, refer to the [Hardware requirements](#) section.

Updated: August 17, 2023

Cluster environment optimization

Redis Enterprise Software uses various algorithms to optimize performance. As part of this process, Redis Enterprise Software examines usage and load to adjust its runtime configuration. Depending on your specific usage and load, Redis Enterprise Software might take some time to adjust for optimal performance.

To ensure optimal performance, you must run your workload several times and for a long duration until performance stabilizes.

Failure detection sensitivity policies

You can optimize your cluster's thresholds and timeouts for different environments using the `failure_detection_sensitivity` cluster policy:

- `high` (previously known as `local-network_watchdog_profile`) – high failure detection sensitivity, lower thresholds, and faster failure detection and failover
- `low` (previously known as `cloud_watchdog_profile`) – low failure detection sensitivity and higher tolerance for latency variance (also called network jitter)

Depending on which policy you choose, Redis Enterprise Software uses different thresholds to make operation-related decisions.

The default policy is high failure detection sensitivity for `local-network` environments. If you are running Redis Enterprise in a cloud environment, you should change the configuration.

Change failure detection sensitivity

To change the cluster's `failure_detection_sensitivity`, run one of the following `rladmin` commands.

- For Redis Enterprise Software version 6.4.2-69 and later, run:

```
rladmin tune cluster failure_detection_sensitivity [ low | high ]
```

- For Redis Enterprise Software version 6.4.2-61 and earlier, run:

```
rladmin tune cluster watchdog_profile [ cloud | local-network ]
```

If Redis Enterprise Software still does not meet your performance expectations after following these instructions, [contact support](#) for help optimizing your environment.

Updated: May 11, 2023

Turn off services to free system memory

The Redis Enterprise Software cluster nodes host a range of services that support the cluster processes. In most deployments, either all of these services are required, or there are enough memory resources on the nodes for the database requirements.

In a deployment with limited memory resources, certain services can be disabled from API endpoint to free system memory or using the `rladmin` command. Before you turn off a service, make sure that your deployment does not depend on that service. After you turn off a service, you can re-enable in the same way.

The services that you can turn off are:

- RS Admin Console - `cm_server`
- Logs in CSV format - `stats_archiver`
- [LDAP authentication](#) - `saslauthd`
- [Discovery service](#) - `mdns_server,pdns_server`
- [Active-Active databases](#) - `crdb_coordinator,crdb_worker`
- Alert Manager - `alert_mgr` (For best results, disable only if you have an alternate alert system.)

To turn off a service with the `rladmin cluster config` command, use the `services` parameter and the name of the service, followed by `disabled`.

```
rladmin cluster config
    [ services <service_name> <enabled | disabled> ]
```

To turn off a service with the API, use the [PUT /v1/services_configuration](#) endpoint with the name of the service and the operating mode (enabled/disabled) in JSON format.

For example:

- To turn off the Redis Enterprise admin console, use this PUT request:

```
PUT https://[host][:9443]/v1/cluster/services_configuration
{
  "cm_server": {
    "operating_mode": "disabled"
  }
}
```

- To turn off the CRDB services and enable the `stats_archiver` for cluster component statistics, use this PUT request:

```
PUT https://[host][:9443]/v1/cluster/services_configuration
'{
  "crdb_coordinator": {
    "operating_mode": "disabled"
  },
  "crdb_worker": {
    "operating_mode": "disabled"
  },
  "stats_archiver": {
    "operating_mode": "enabled"
  }
}'
```

Updated: February 14, 2023

Redis OSS Cluster API Architecture

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.

You can use Redis OSS Cluster API along with other Redis Enterprise Software high availability to get high performance with low latency and let applications stay current with cluster topology changes, including add node, remove node, and node failover.

For more about working with the OSS Cluster API, see [Using the OSS Cluster API](#).

Updated: August 31, 2022

Use the WAIT command for strong consistency

Redis Enterprise Software comes with the ability to replicate data to another replica for high availability and persist in-memory data on disk permanently for durability. With the WAIT command, you can control the consistency and durability guarantees for the replicated and persisted database.

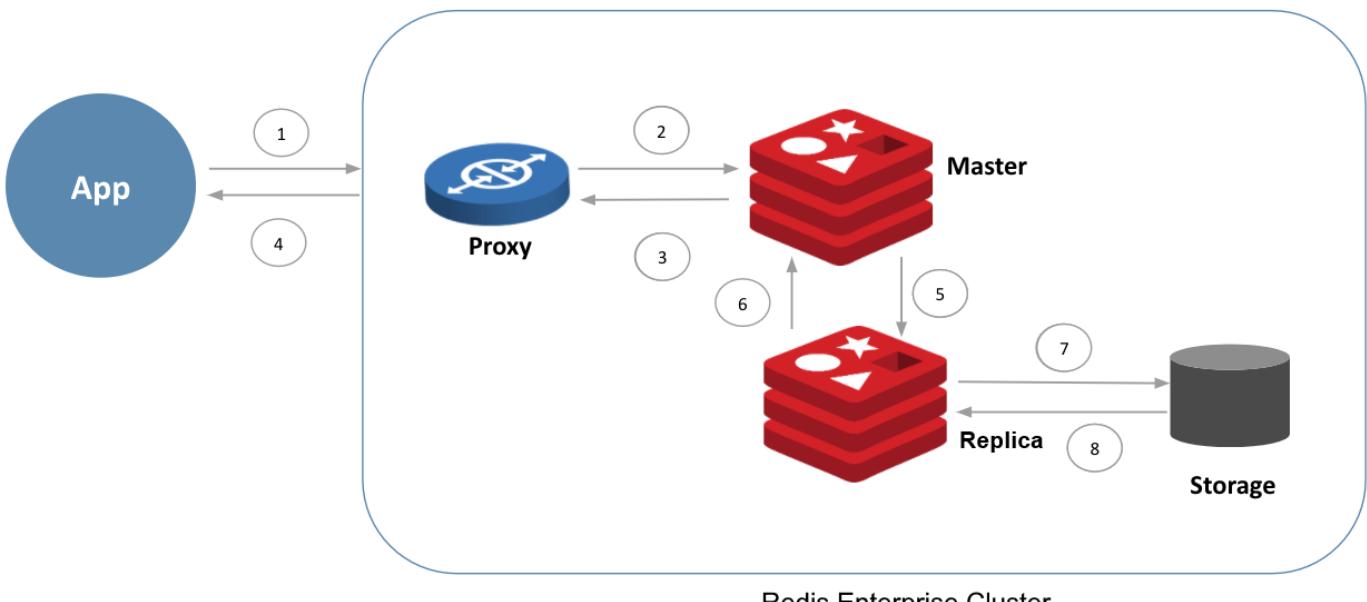
Any updates that are issued to the database are typically performed with the following flow shown below;

1. Application issues a write,
2. Proxy communicates with the correct master “shard” in the system that contains the given key,
3. The acknowledgment is sent to proxy once the write operation completes
4. The proxy sends the acknowledgment back to the application.

Independently, the write is communicated from master to replica and replication acknowledges the write back to the master. These are steps 5 and 6.

Independently, the write to a replica is also persisted to disk and acknowledged within the replica. These are steps 7 and 8.

Weak Consistency

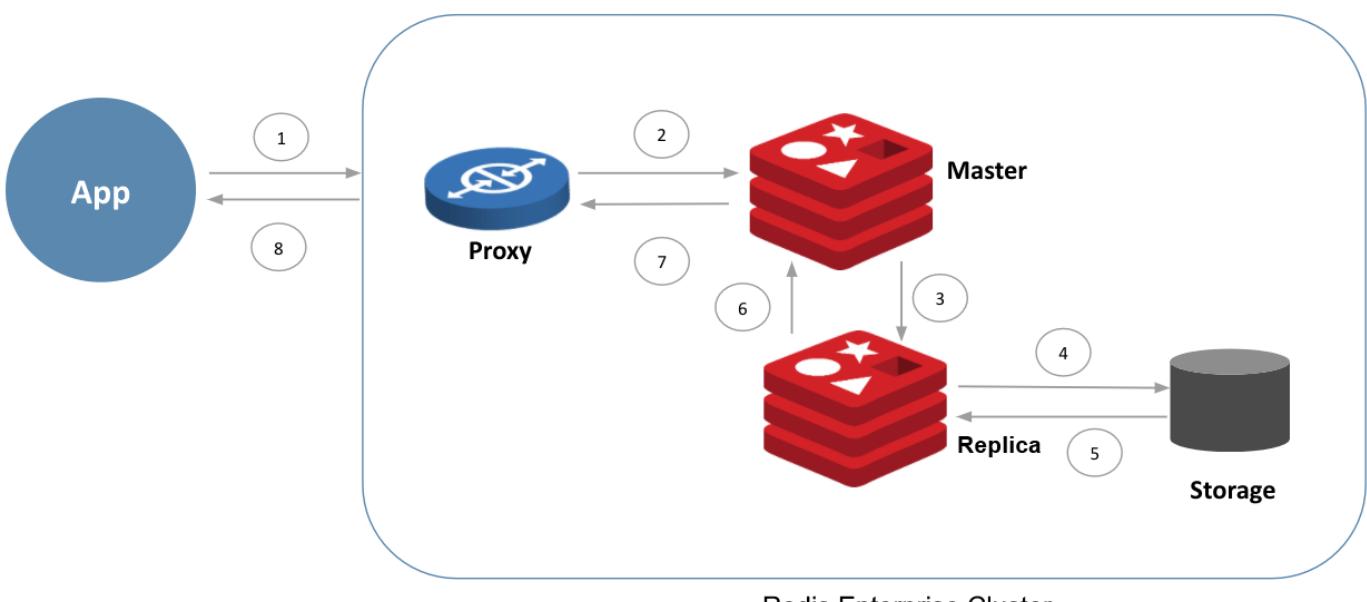


With the WAIT command, applications can ask to wait for acknowledgments only after replication or persistence is confirmed on the replica. The flow of a write operation with the WAIT command is shown below:

1. Application issues a write,
2. Proxy communicates with the correct master "shard" in the system that contains the given key,
3. Replication communicated the update to the replica shard.
4. Replica persists the update to disk (assuming AOF every write setting is selected).
5. The acknowledgment is sent back from the replica all the way to the proxy with steps 5 to 8.

With this flow, the application only gets the acknowledgment from the write after durability is achieved with replication to the replica and to the persistent storage.

Strong Consistency



With the WAIT command, applications can have a guarantee that even under a node failure or node restart, an acknowledged write is recorded.

See the WAIT command for details on the new durability and consistency options.

Maintenance mode for cluster nodes

Use maintenance mode to prevent data loss during hardware or operating system maintenance on Redis Enterprise servers. When maintenance mode is on, all shards move off of the node under maintenance and migrate to another available node.

Activate maintenance mode

When you activate maintenance mode, Redis Enterprise does the following:

1. Checks whether a shut down of the node will cause quorum loss. If so, maintenance mode will not turn on.

Maintenance mode does not protect against quorum loss. If you activate maintenance mode for the majority of nodes in a cluster and restart them simultaneously, quorum is lost, which can lead to data loss.

2. Takes a snapshot of the node configuration in order to record the shard and endpoint configuration of the node.
3. Marks the node as a quorum node to prevent shards and endpoints from migrating to it.

At this point, `radmin status` displays the node's shards field in yellow, which indicates that shards cannot migrate to the node.

```
radmin> node 2 maintenance_mode on
Performing maintenance_on action on node:2: 0%
created snapshot NodeSnapshot<name=maintenance_mode_2019-04-10_13-33-50,time=None,node_uid=2>

node:2 will not accept any more shards
Performing maintenance_on action on node:2: 100%
OK
radmin> status
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES RAM AVAILABLE_RAM VERSION STATUS
*node:1 master 172.17.0.2 rp1 4/100 2 1.84GB/6.78GB 537.77MB/5.56GB 100.0.2-3572 OK
node:2 slave 172.17.0.3 rp2 0/0 2 1.84GB/6.78GB 0KB/0KB 100.0.2-3572 OK
node:3 slave 172.17.0.4 rp3 0/100 2 1.84GB/6.78GB 632.59MB/5.56GB 100.0.2-3572 OK
```

4. Migrates shards and binds endpoints to other nodes, when space is available.

Maintenance mode does not demote a master node by default. The cluster elects a new master node when the original master node restarts.

Add the `demote_node` option to the `radmin` command to [demote a master node](#) when you activate maintenance mode.

To activate maintenance mode for a node, run the following command:

```
radmin node <node_id> maintenance_mode on
```

You can start server maintenance if:

- All shards and endpoints have moved to other nodes
- Enough nodes are still online to maintain quorum

Prevent replica shard migration

If you do not have enough resources available to move all of the shards to other nodes, you can turn maintenance mode on without migrating the replica shards.

If you prevent replica shard migration, the shards remain on the node during maintenance.

If the maintenance node fails in this case, the master shards do not have replica shards to maintain data redundancy and high availability.

To activate maintenance mode without replica shard migration, run:

```
radmin node <node_id> maintenance_mode on keep_slave_shards
```

Demote a master node

If maintenance might affect connectivity to the master node, you can demote the master node when you activate maintenance mode. This lets the cluster elect a new master node.

To demote a master node when activating maintenance mode, run:

```
radmin node <node_id> maintenance_mode on demote_node
```

Verify maintenance mode activation

To verify maintenance mode for a node, use `radmin status` and review the node's shards field. If that value is displayed in yellow (shown earlier), then the node is in maintenance mode.

Avoid activating maintenance mode when it is already active. Maintenance mode activations stack. If you activate maintenance mode for a node that is already in maintenance mode, you will have to deactivate maintenance mode twice in order to restore full functionality.

Deactivate maintenance mode

When you deactivate maintenance mode, Redis Enterprise:

1. Loads a [specified snapshot](#) or defaults to the latest snapshot.
2. Unmarks the node as a quorum node to allow shards and endpoints to migrate to the node.
3. Restores the shards and endpoints that were in the node at the time of the snapshot.
4. Deletes the snapshot.

To deactivate maintenance mode after server maintenance, run:

```
radmin node <node_id> maintenance_mode off
```

By default, a snapshot is required to deactivate maintenance mode. If the snapshot cannot be restored, deactivation is cancelled and the node remains in maintenance mode. In such events, it may be necessary to [reset node status](#).

Specify a snapshot

Redis Enterprise saves a snapshot of the node configuration every time you turn on maintenance mode. If multiple snapshots exist, you can restore a specific snapshot when you turn maintenance mode off.

To get a list of available snapshots, run:

```
radmin node <node_id> snapshot list
```

To specify a snapshot when you turn maintenance mode off, run:

```
radmin node <node_id> maintenance_mode off snapshot_name <snapshot_name>
```



Note: If an error occurs when you turn on maintenance mode, the snapshot is not deleted. When you rerun the command, use the snapshot from the initial attempt since it contains the original state of the node.

Skip shard restoration

You can prevent the migrated shards and endpoints from returning to the original node after you turn off maintenance mode.

To turn maintenance mode off and skip shard restoration, run:

```
radmin node <node_id> maintenance_mode off skip_shards_restore
```

Reset node status

In extreme cases, you may need to reset a node's status. Run the following commands to do so:

```
$ radmin tune node <node_id> max_listeners 100  
$ radmin tune node <node_id> quorum_only disabled
```

Use these commands with caution. For best results, contact Support before running these commands.

Cluster status example

This example shows how the output of `rladmin status` changes when you turn on maintenance mode for a node.

The cluster status before turning on maintenance mode:

```
redislabs@rp1_node1:/opt$ rladmin status
CLUSTER NODES:
NODE:ID  ROLE      ADDRESS      EXTERNAL_ADDRESS      HOSTNAME      SHARDS
*node:1  master    172.17.0.2          rp1_node1      2/100
node:2   slave     172.17.0.4          rp3_node1      2/100
node:3   slave     172.17.0.3          rp2_node1      0/100
```

The cluster status after turning on maintenance mode:

```
redislabs@rp1_node1:/opt$ rladmin node 2 maintenance_mode on
Performing maintenance_on action on node:2: 0%
created snapshot NodeSnapshot<name=maintenance_mode_2019-03-14_09-50-59,time=None,node_uid=2>

node:2 will not accept any more shards
Performing maintenance_on action on node:2: 100%
OK
redislabs@rp1_node1:/opt$ rladmin status
CLUSTER NODES:
NODE:ID  ROLE      ADDRESS      EXTERNAL_ADDRESS      HOSTNAME      SHARDS
*node:1  master    172.17.0.2          rp1_node1      2/100
node:2   slave     172.17.0.4          rp3_node1      0/0
node:3   slave     172.17.0.3          rp2_node1      2/100
```

After turning on maintenance mode for node 2, Redis Enterprise saves a snapshot of its configuration and then moves its shards and endpoints to node 3.

Now node 2 has 0/0 shards because shards cannot migrate to it while it is in maintenance mode.

Maintenance mode REST API

You can also turn maintenance mode on or off via [REST API requests to POST /nodes/{node_uid}/actions/{action}](#).

Activate maintenance mode (REST API)

Use `POST /nodes/{node_uid}/actions/maintenance_on` to activate maintenance mode:

```
POST https://[host][:port]/v1/nodes/<node_id>/actions/maintenance_on
'{"keep_slave_shards":true}'
```

The `keep_slave_shards` boolean flag [prevents replica shard migration](#) when set to `true`.

The `maintenance_on` request returns a JSON response body:

```
{
  "status": "queued",
  "task_id": "<task-id-guid>"
}
```

Deactivate maintenance mode (REST API)

Use `POST /nodes/{node_uid}/actions/maintenance_off` deactivate maintenance mode:

```
POST https://[host][:port]/v1/nodes/<node_id>/actions/maintenance_off
'{"skip_shards_restore":false}'
```

The `skip_shards_restore` boolean flag allows the `maintenance_off` action to [skip shard restoration](#) when set to `true`.

The `maintenance_off` request returns a JSON response body:

```
{  
    "status": "queued",  
    "task_id": "<task-id-guid>"  
}
```

Track action status

You can send a request to [GET /nodes/{node_uid}/actions/{action}](https://<hostname>/v1/nodes/<node_uid>/actions/<action>) to track the `status` of the `maintenance_on` and `maintenance_off` actions.

This request returns the status of the `maintenance_on` action:

```
GET https://<hostname>:9443/v1/nodes/<node_id>/actions/maintenance_on
```

The response body:

```
{  
    "status": "completed",  
    "task_id": "38c7405b-26a7-4379-b84c-cab4b3db706d"  
}
```

Updated: November 14, 2022

Recover a failed cluster

When a Redis Enterprise Software cluster fails, you must use the cluster configuration file and database data to recover the cluster.

 | **Note:** For cluster recovery in a Kubernetes deployment, go to: [Recover a Redis Enterprise cluster on Kubernetes](#).

Cluster failure can be caused by:

- A hardware or software failure that causes the cluster to be unresponsive to client requests or administrative actions.
- More than half of the cluster nodes lose connection with the cluster, resulting in quorum loss.

To recover a cluster and re-create it as it was before the failure you must restore the cluster configuration (`ccs-redis.rdb`) to the cluster nodes. To restore the data that was in the databases to databases in the new cluster you must restore the database persistence files (backup, AOF, or snapshot files) to the databases. These files are stored in the [persistent storage location](#).

The cluster recovery process includes:

1. Install RS on the nodes of the new cluster.
2. Mount the persistent storage with the recovery files from the original cluster to the nodes of the new cluster.
3. Recover the cluster configuration on the first node in the new cluster.
4. Join the remaining nodes to the new cluster.
5. [Recover the databases](#).

Prerequisites

- We recommend that you recover the cluster to clean nodes. If you use the original nodes, make sure there are no Redis processes running on any nodes in the new cluster.
- We recommend that you use clean persistent storage drives for the new cluster. If you use the original storage drives, make sure that you backup the files on the original storage drives to a safe location.
- Identify the cluster configuration file that you want to use as the configuration for the recovered cluster. The cluster configuration file is `/css/ccs-redis.rdb` on the persistent storage for each node.

Recovering the cluster

1. (Optional) If you want to recover the cluster to the original cluster nodes, uninstall RS from the nodes.
2. Install RS on the new cluster nodes.

Do not configure the cluster nodes (`radmin cluster create` in the CLI or `Setup` in the admin console).

The new servers must have the same basic hardware and software configuration as the original servers, including:

- The same number of nodes
- At least the same amount of memory
- The same RS version
- The same installation user and paths

 **Note:** The cluster recovery can fail if these requirements are not met.

3. Mount the persistent storage drives with the recovery files to the new nodes. These drives must contain the cluster configuration backup files and database persistence files.

 **Note:** Make sure that the user `redislabs` has permissions to access the storage location of the configuration and persistence files on each of the nodes.

If you use local persistent storage, place all of the recovery files on each of the cluster nodes.

4. To recover the cluster configuration from the original cluster to the first node in the new cluster, from the `radmin` command-line interface (CLI):

```
radmin cluster recover filename [ <persistent_path> | <ephemeral_path> ]<filename> node_uid <node_uid>  
rack_id <rack_id>
```

+ [Command syntax](#)

<filename> - The full path of the old cluster configuration file in the persistent storage. The cluster configuration file is `/css/ccs-redis.rdb`.
<node_uid> - The id of the node, in this case 1.
<persistent_path> (optional) - The location of the [persistent storage](#) in the new node.
<ephemeral_path> (optional) - The location of the [ephemeral storage](#) in the new node.
<rack_id> (optional) - If [rack-zone awareness](#) was enabled in the cluster, you can use this parameter to override the rack ID value that was set for the node with ID 1 with a new rack ID. Otherwise, the node gets the same rack ID as the original node.

For example:

```
radmin cluster recover filename /tmp/persist/css/ccs-redis.rdb node_uid 1 rack_id 5
```

When the recovery command succeeds, this node is configured as the node from the old cluster that has ID 1.

5. To join the remaining servers to the new cluster, run `radmin cluster join` from each new node:

```
radmin cluster join [ nodes <cluster_member_ip_address> | name <cluster_FQDN> ] username <username>  
password <password> replace_node <node_id>
```

+ [Command syntax](#)

nodes - The IP address of a node in the cluster that this node is joining.
name - The [FQDN name](#) of the cluster this node is joining.
username - The email address of the cluster administrator.
password - The password of the cluster administrator.
replace_node - The ID of the node that this node replaces from the old cluster.
persistent_path (optional) - The location of the [persistent storage](#) in the new node.
ephemeral_path (optional) - The location of the [ephemeral storage](#) in the new node.
rack_id (optional) - If [rack-zone awareness](#) was enabled in the cluster, use this parameter to set the rack ID to be the same as the rack ID of the old node. You can also change the value of the rack ID by providing a different value and using the `override_rack_id` flag.

For example:

```
radmin cluster join nodes 10.142.0.4 username admin@example.com password mysecret replace_node 2
```

You can run the `radmin status` command to verify that the recovered nodes are now active, and that the databases are pending recovery.



Note: Make sure that you update your [DNS records](#) with the IP addresses of the new nodes.

After the cluster is recovered, you must [recover the databases](#).

Updated: August 17, 2023

Remove a cluster node

There are various reasons why you may want to remove a node in Redis Enterprise Software:

- You no longer need the extra capacity, meaning you want to permanently remove the node.
- You would like to replace a faulty node with a healthy node.
- You would like to replace a healthy node with a different node.

The following section explains how each of these actions can be achieved, as well as their impact and considerations.

You can configure [email alerts from the cluster](#) to notify you of cluster changes, including when a node is removed.

Make sure to read through these explanations thoroughly before taking any action.

Permanently remove a node

Permanently removing a node means you are decreasing cluster capacity. Before trying to remove a node, make sure that the cluster has enough capacity for all resources without that node, otherwise you cannot remove the node.

If there is not enough capacity in the cluster to facilitate removing the node, you can either delete databases or add another node instead of the one you would like to remove.

During the removal process, the cluster migrates all resources from the node being removed to other nodes in the cluster. In order to ensure database connectivity, and database high availability (when replication is enabled), the cluster first creates replacement shards or endpoints on one of the other nodes in the cluster, initiates failover as needed, and only then removes the node.

If a cluster has only two nodes (which is not recommended for production deployments) and some databases have replication enabled, you cannot remove a node.

Replace a faulty node

If the cluster has a faulty node that you would like to replace, you only need to add a new node to the cluster. The cluster recognizes the existence of a faulty node and automatically replaces the faulty node with the new node.

For guidelines, refer to [Replacing a faulty node](#).

Replace a healthy node

If you would like to replace a healthy node with a different node, you must first add the new node to the cluster, migrate all the resources from the node you would like to remove, and only then remove the node.

For further guidance, refer to [adding a new node to a cluster](#).

You can migrate resources by using the `radmin` command-line interface (CLI). For guidelines, refer to [radmin command-line interface \(CLI\)](#).



Note: The [DNS records](#) must be updated each time a node is added or replaced.

Remove a node

To remove a node using the admin console:

1. If you are using the new admin console, switch to the legacy admin console.



2. Click **Remove** at the top of the **Node** page for the node to be removed.
3. Approve the action.
4. Redis Enterprise Software examines the node and the cluster and takes the actions required to remove the node.
5. At any point, you can click the **Abort** button to stop the process. When aborted, the current internal action is completed, and then the process stops.
6. Once the process finishes, the node is no longer shown in the UI.

To remove a node using the REST API, use `POST /v1/nodes/<node_id>/actions/remove` with the following JSON data and the "Content-Type: application/json" header.

For example:

```
POST https://[host][:port]/v1/nodes/<node_id>/actions/remove  
"{}"
```



Note: If you need to add a removed node back to the cluster, you must [uninstall](#) and [reinstall](#) the software on that node.

Updated: August 17, 2023

Replace a cluster node

A failed node will appear as Down (🔴) in the **Nodes** list.

To replace a failed node:

1. Acquire a new node identical to the old one.
2. Install and configure Redis Enterprise Software on the node. See [Install and setup](#) for more information.



Note: If you are using [Auto Tiering](#), make sure the required flash storage is set up on this new node.

3. [Add the node](#) to the cluster. Make sure the new node has as much available memory as the faulty node.

If the new node does not have enough memory, you will be prompted to add a node with enough memory.

4. A message will appear informing you that the cluster has a faulty node and that the new node will replace the faulty node.



Note:

- If there is a faulty node in the cluster to which you are adding a node, Redis Enterprise Software will use the new node to replace the faulty one.
- Any existing [DNS records](#) must be updated each time a node is added or replaced.

Updated: August 17, 2023

Logging events

Management actions performed with Redis Enterprise are logged to make sure system management tasks are appropriately performed or monitored by administrators and for compliance with regulatory standards.

Log entries contain the following information:

1. Who performed the action?
2. What exactly was the performed action?
3. When was the action performed?
4. Did the action succeed or not?

To get the list of logged events, you can use the REST API or the **Logs** screen in the UI. The **Logs** screen displays the system and user events regarding alerts, notifications, and configuration.

Time	Originator	Source	Type	Description
8/1/2023 8:56:55 PM	system	Cluster	Notification	Database activated. Database: Database-01-Aug-23-15-56PM
8/1/2023 8:56:53 PM	Administrator [ra...]	Bdb		descriptionMap.bdb_bdb_create_request
8/1/2023 8:56:28 PM	system	Database-28-Jul-...	Configuration	Clustering - number of shards changed from 1 to 2.
8/1/2023 8:56:27 PM	Administrator [ra...]	Bdb		descriptionMap.bdb_bdb_update_request
8/1/2023 8:56:27 PM	Administrator [ra...]	Bdb		descriptionMap.bdb_bdb_update_request
8/1/2023 8:56:27 PM	Administrator [ra...]	Bdb		descriptionMap.bdb_bdb_update_request
8/1/2023 8:54:24 PM	system	User	Notification	Login succeeded (username: Administrator)
8/1/2023 2:47:32 PM	system	User	Notification	Login succeeded (username: Administrator)

You can use the **Logs** screen to review what actions a user has performed, such as editing a database's configuration.

- [Redis slow log](#)
- [rsyslog logging](#)

View logs in the UI

Redis Enterprise provides log files for auditing cluster management actions and troubleshooting. You can view these logs in the UI and on the host operating system.

To view event logs in the new Cluster Manager UI, go to **Cluster > Logs**.

View logs on the server

Server logs can be found by default in the directory `/var/opt/redislabs/log/`.

These log files are used by the Redis support team to troubleshoot issues. The logs you will most frequently interact with is 'event_log.log'. This log file is where logs of configuration actions within Redis are stored and is useful to determine events that occur within Redis Enterprise.

Configure log timestamps

Redis Enterprise allows you to configure log timestamps. To configure log timestamps in the new Cluster Manager UI:

1. Go to **Cluster > Configuration > General**.
2. Change the **Time zone** for the logs based on your location.

Log security

Redis Enterprise comes with a set of logs on the server and available through the user interface to assist users in investigating actions taken on the server and to troubleshoot issues.

Send logs to a remote logging server

Redis Enterprise sends logs to syslog by default. You can send these logs to a remote logging server by configuring syslog.

To do this, modify the syslog or rsyslog configuration on your operating system to send logs in the \$logdir directory (/var/opt/redislabs/log in default installations) to a remote monitoring server of your choice.

Log rotation

Redis Enterprise Software's job scheduler runs logrotate every five minutes to examine logs stored on the operating system and rotate them based on the log rotation configuration. You can find the log rotation configuration file at \$pkgconfdir/logrotate.conf as of Redis Enterprise Software version 7.2 (pkgconfdir is /opt/redislabs/config by default, but can be changed in a custom installation).

By default, log rotation occurs when a log exceeds 200 MB. We recommend sending log files to a remote logging server so you can maintain them more effectively.

The following log rotation policy is enabled by default in Redis Enterprise Software, but you can modify it as needed.

```
/var/opt/redislabs/log/*.log {
    su ${osuser} ${osgroup}
    size 200M
    missingok
    copytruncate
    # 2000 is logrotate's way of saying 'infinite'
    rotate 2000
    maxage 7
    compress
    notifempty
    nodateext
    nosharedscripts
    prerotate
        # copy cluster_wd log to another file that will have longer retention
        if [ "\$1" = "/var/opt/redislabs/log/cluster_wd.log" ]; then
            cp -p /var/opt/redislabs/log/cluster_wd.log /var/opt/redislabs/log/cluster_wd.log.long_retention
        fi
    endscript
}
/var/opt/redislabs/log/cluster_wd.log.long_retention {
    su ${osuser} ${osgroup}
    daily
    missingok
    copytruncate
    rotate 30
    compress
    notifempty
    nodateext
}
```

- `/var/opt/redislabs/log/*.log` - logrotate checks the files under the \$logdir directory (/var/opt/redislabs/log/) and rotates any files that end with the extension .log.
- `/var/opt/redislabs/log/cluster_wd.log.long_retention` - The contents of `cluster_wd.log` is copied to `cluster_wd.log.long_retention` before rotation, and this copy is kept for longer than normal (30 days).
- `size 200M` - Rotate log files that exceed 200 MB.
- `missingok` - If there are missing log files, do nothing.
- `copytruncate` - Truncate the original log file to zero sizes after creating a copy.
- `rotate 2000` - Keep up to 2000 (effectively infinite) log files.

- compress - gzip log files.
- maxage 7 - Keep the rotated log files for 7 days.
- notifempty - Don't rotate the log file if it is empty.



Note: For large scale deployments, you might need to rotate logs at faster intervals than daily. You can also use a cronjob or external vendor solutions.

Updated: August 17, 2023

rsyslog logging

This document explains the structure of Redis Enterprise Software log entries in rsyslog and how to use these log entries to identify events.



Note: You can also [secure your logs](#) with a remote logging server and log rotation.

Log concepts

Redis Enterprise Software logs information from a variety of components in response to actions and events that occur within the cluster.

In some cases, a single action, such as removing a node from the cluster, may actually consist of several events. These actions may generate multiple log entries.

All log entries displayed in the admin console are also written to syslog. You can configure rsyslog to monitor syslog. Enabled alerts are logged to syslog and appear with other log entries.

Types of log entries

Log entries are categorized into events and alerts. Both types of entries appear in the logs, but alert log entries also include a boolean "state" parameter that indicates whether the alert is enabled or disabled.

Log entries include information about the specific event that occurred. See the log entry tables for [clusters](#), [databases](#), [nodes](#), and [users](#) for more details.

Severity

You can also configure rsyslog to add other information, such as the event severity.

Since rsyslog entries do not include severity by default, you can follow these steps to enable it:

1. Add the following line to /etc/rsyslog.conf:

```
$template TraditionalFormatWithPRI, "%pri-text%: %timegenerated% %HOSTNAME% %syslogtag%%msg:::drop-last-1f%\n"
```

2. Modify \$ActionFileDefaultTemplate to use your new template \$ActionFileDefaultTemplateTraditionalFormatWithPRI

3. Save these changes and restart rsyslog to apply them

You can see the log entries for alerts and events in the /var/log/messages file.

Command components:

- %pri_text% adds the severity
- %timegenerated% adds the timestamp
- %HOSTNAME% adds the machine name
- %syslogtag% adds the Redis Enterprise Software message. See the [log entry structure](#) section for more details.
- %msg:::drop last 1f% removes duplicated log entries

Log entry structure

The log entries have the following basic structure:

```
event_log[<process id>]:{<list of key-value pairs in any order>}
```

- **event_log**: Plain static text is always shown at the beginning of the entry.
- **process id**: The ID of the logging process
- **list of key-value pairs in any order**: A list of key-value pairs that describe the specific event. They can appear in any order. Some key-value pairs are always shown, and some appear depending on the specific event.
 - **Key-value pairs that always appear**:
 - "type": A unique code name for the logged event. For the list of codenames, see the logged events and alerts tables for [clusters](#), [databases](#), [nodes](#), and [users](#).
 - "object": Defines the object type and ID (if relevant) of the object this event relates to, such as cluster, node with ID, BDB with ID, etc. Has the format of <object_type>[:<id>].
 - "time": Unix epoch time but can be ignored in this context.
 - **Key-value pairs that might appear depending on the specific entry**:
 - "state": A boolean where true means the alert is enabled, and false means the alert is disabled. This is only relevant for alert log entries.
 - "global_threshold": The value of a threshold for alerts related to cluster or node objects.
 - "threshold": The value of a threshold for [alerts related to a BDB object](#).

Log entry samples

This section provides examples of log entries that include the [rsyslog configuration](#) to add the severity, timestamp, and machine name.

Ephemeral storage passed threshold

"Alert on" log entry sample

```
daemon.warning: Jun 14 14:49:20 node1 event_log[3464]:  
{  
  "storage_util": 90.061643120001,  
  "global_threshold": "70",  
  "object": "node:1",  
  "state": true,  
  "time": 1434282560,  
  "type": "ephemeral_storage"  
}
```

In this example, the storage utilization on node 1 reached the value of ~90%, which triggered the alert for "Ephemeral storage has reached 70% of its capacity."

Log entry components:

- `daemon.warning` - Severity of entry is warning
- Jun 14 14:49:20 - The timestamp of the event
- `node1`: Machine name
- `event_log` - Static text that always appears
- [3464] - Process ID
- "storage_util":90.061643120001 - Current ephemeral storage utilization
- "global_threshold": "70" - The user-configured threshold above which the alert is raised
- "object": "node:1" - The object related to this alert
- "state":true - Current state of the alert
- "time":1434282560 - Can be ignored
- "type": "ephemeral_storage" - The code name of this specific event. See [logged node alerts and events](#) for more details.

"Alert off" log entry sample

```
daemon.info: Jun 14 14:51:35 node1 event_log[3464]:  
{  
  "storage_util":60.051723520008,  
  "global_threshold": "70",  
  "object": "node:1",  
  "state":false,  
  "time": 1434283480,  
  "type": "ephemeral_storage"  
}
```

This log entry is an example of when the alert for the node with ID 1 "Ephemeral storage has reached 70% of its capacity" has been turned off as result of storage utilization reaching the value of ~60%.

Log entry components:

- daemon.info - Severity of entry is info
- Jun 14 14:51:35 - The timestamp of the event
- node1 - Machine name
- event_log - Static text that always appears
- [3464] - Process ID
- "storage_util":60.051723520008 - Current ephemeral storage utilization
- "global_threshold": "70" - The user configured threshold above which the alert is raised (70% in this case)
- "object": "node:1" - The object related to this alert
- "state": false - Current state of the alert
- "time": 1434283480 - Can be ignored
- "type": "ephemeral_storage" - The code name identifier of this specific event. See [logged node alerts and events](#) for more details.

Odd number of nodes with a minimum of three nodes alert

"Alert on" log entry sample

```
daemon.warning: Jun 14 15:25:00 node1 event_log[8310]:  
{  
    "object": "cluster",  
    "state": true,  
    "time": 1434284700,  
    "node_count": 1,  
    "type": "even_node_count"  
}
```

This log entry is an example of when the alert for "True high availability requires an odd number of nodes with a minimum of three nodes" has been turned on as result of the cluster having only one node.

Log entry components:

- daemon.warning - Severity of entry is warning
- Jun 14 15:25:00 - The timestamp of the event
- node1 - Machine name
- event_log - Static text that always appears
- [8310] - Process ID
- "object": "cluster" - The object related to this alert
- "state": true - Current state of the alert
- "time": 1434284700 - Can be ignored
- "node_count": 1 - The number of nodes in the cluster
- "type": "even_node_count" - The code name identifier of this specific event. See [logged cluster alerts and events](#) for more details.

"Alert off" log entry sample

```
daemon.warning: Jun 14 15:30:40 node1 event_log[8310]:  
{  
    "object": "cluster",  
    "state": false,  
    "time": 1434285200,  
    "node_count": 3,  
    "type": "even_node_count"  
}
```

This log entry is an example of when the alert for "True high availability requires an odd number of nodes with a minimum of three nodes" has been turned off as result of the cluster having 3 nodes.

Log entry components:

- daemon.warning - Severity of entry is warning
- Jun 14 15:30:40 - The timestamp of the event
- node1 - Machine name
- event_log - Static text that always appears
- [8310] - Process ID
- "object": "cluster" - The object related to this alert

- "state":false - Current state of the alert
- "time":1434285200 - Can be ignored
- "node_count":3 - The number of nodes in the cluster
- "type":"even_node_count" - The code name of this specific event. See [logged cluster alerts and events](#) for more details.

Node has insufficient disk space for AOF rewrite

"Alert on" log entry sample

```
daemon.err: Jun 15 13:51:23 node1 event_log[34252]:
{
    "used": 23457188,
    "missing": 604602126,
    "object": "node:1",
    "free": 9867264,
    "needed": 637926578,
    "state": true,
    "time": 1434365483,
    "disk": 705667072,
    "type": "insufficient_disk_aofrw"
}
```

This log entry is an example of when the alert for "Node has insufficient disk space for AOF rewrite" has been turned on as result of not having enough persistent storage disk space for AOF rewrite purposes. It is missing 604602126 bytes.

Log entry components:

- daemon.err - Severity of entry is error
- Jun 15 13:51:23 - The timestamp of the event
- node1 - Machine name
- event_log - Static text that always appears
- [34252] - Process ID
- "used":23457188 - The amount of disk space in bytes currently used for AOF files
- "missing":604602126 - The amount of disk space in bytes that is currently missing for AOF rewrite purposes
- "object":"node:1" - The object related to this alert
- "free":9867264 - The amount of disk space in bytes that is currently free
- "needed":637926578 - The amount of total disk space in bytes that is needed for AOF rewrite purposes
- "state":true - Current state of the alert
- "time":1434365483 - Can be ignored
- "disk":705667072 - The total size in bytes of the persistent storage
- "type": "insufficient_disk_aofrw" - The code name of this specific event. See [logged node alerts and events](#) for more details.

"Alert off" log entry sample

```
daemon.info: Jun 15 13:51:11 node1 event_log[34252]:
{
    "used": 0, "missing": -21614592,
    "object": "node:1",
    "free": 21614592,
    "needed": 0,
    "state": false,
    "time": 1434365471,
    "disk": 705667072,
    "type": "insufficient_disk_aofrw"
}
```

Log entry components:

- daemon.info - Severity of entry is info
- Jun 15 13:51:11 - The timestamp of the event
- node1 - Machine name
- event_log - Static text that always appears
- [34252] - Process ID
- "used":0 - The amount of disk space in bytes currently used for AOF files
- "missing": -21614592 - The amount of disk space in bytes that is currently missing for AOF rewrite purposes. In this case, it is not missing because the number is negative.
- "object":"node:1" - The object related to this alert

- "free": 21614592 - The amount of disk space in bytes that is currently free
- "needed": 0 - The amount of total disk space in bytes that is needed for AOF rewrite purposes. In this case, no space is needed.
- "state": false - Current state of the alert
- "time": 1434365471 - Can be ignored
- "disk": 705667072 - The total size in bytes of the persistent storage
- "type": "insufficient_disk_aofrw" - The code name of this specific event. See [logged node alerts and events](#) for more details.

Updated: August 31, 2022

Cluster alert and event logs

The following cluster alerts and events can appear in syslog.

UI alerts

Logged alerts that appear in the UI

Alert code name	Alert as shown in the UI	Severity	Notes
even_node_count	True high availability requires an odd number of nodes with a minimum of three nodes	true: warning false: info	
inconsistent_redis_sw	Not all databases are running the same open source version	true: warning false: info	
inconsistent_rl_sw	Not all nodes in the cluster are running the same Redis Enterprise Cluster version	true: warning false: info	
internal_bdb	Issues with internal cluster databases	true: warning false: info	
multiple_nodes_down	Multiple cluster nodes are down - this might cause data loss	true: warning false: info	
ram_overcommit	Cluster capacity is less than total memory allocated to its databases	true: error false: info	
too_few_nodes_for_replication	Database replication requires at least two nodes in cluster	true: warning false: info	

UI events

Logged events that appear in the UI

Event code name	Event as shown in the UI	Severity	Notes
node_joined	Node joined	info	
node_remove_abort_completed	Node removed	info	The remove node is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
node_remove_abort_failed	Node removed	error	The remove node is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
node_remove_completed	Node removed	info	The remove node is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
node_remove_failed	Node removed	error	The remove node is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.

Event code name	Event as shown in the UI	Severity	Notes
rebalance_abort_completed	Nodes rebalanced	info	The nodes rebalance is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
rebalance_abort_failed	Nodes rebalanced	error	The nodes rebalance is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
rebalance_completed	Nodes rebalanced	info	The nodes rebalance is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.
rebalance_failed	Nodes rebalanced	error	The nodes rebalance is a process that can fail and can also be cancelled. If cancelled, the cancellation process can succeed or fail.

Non-UI events

Logged events that do not appear in the UI

Event code name	Severity	Notes
cluster_updated	info	Indicates that cluster settings have been updated
license_added	info	
license_deleted	info	
license_updated	info	

Updated: August 31, 2022

Database alert and event logs

The following database (BDB) alerts and events can appear in syslog.

UI alerts

Logged alerts that appear in the UI

Alert code name	Alert as shown in the UI	Severity	Notes
backup_delayed	Periodic backup has been delayed for longer than minutes	true: warning false: info	Has threshold parameter in the data section of the log entry.
high_latency	Latency is higher than msec	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
high_syncer_lag	Replica of - sync lag is higher than seconds	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
high_throughput	Throughput is higher than RPS (requests per second)	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
low_throughput	Throughput is lower than RPS (requests per second)	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
ram_dataset_overhead	RAM Dataset overhead in a shard has reached % of its RAM limit	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
ram_values	Percent of values in a shard's RAM is lower than % of its key count	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
shard_num_ram_values	Number of values in a shard's RAM is lower than values	true: warning false: info	Has threshold parameter in the key/value section of the log entry.

Alert code name	Alert as shown in the UI	Severity	Notes
size	Dataset size has reached % of the memory limit	true: warning false: info	Has threshold parameter in the key/value section of the log entry.
syncer_connection_error	Replica of - database unable to sync with source	error	
syncer_general_error	Replica of - database unable to sync with source	error	

Non-UI events

Logged events that do not appear in the UI

Event code name	Severity	Notes
authentication_err	error	Replica of - Error authenticating with the source database
backup_failed	error	
backup_started	info	
backup_succeeded	info	
bdb_created	info	
bdb_deleted	info	
bdb_updated	info	Indicates that a BDB configuration has been updated
compression_unsup_err	error	Replica of - Compression not supported by sync destination
crossslot_err	error	Replica of - Sharded destination does not support operation executed on source
export_failed	error	
export_started	info	
export_succeeded	info	
import_failed	error	
import_started	info	
import_succeeded	info	
oom_err	error	Replica of - Replication source/target out of memory

Updated: August 31, 2022

Node alert and event logs

The following node alerts and events can appear in syslog.

UI alerts

Logged alerts that appear in the UI

Alert code name	Alert as shown in the UI	Severity	Notes
aof_slow_disk_io	Redis performance is degraded as result of disk I/O limits	true: error false: info	
cpu_utilization	CPU utilization has reached %	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.
ephemeral_storage	Ephemeral storage has reached % of its capacity	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.

Alert code name	Alert as shown in the UI	Severity	Notes
failed	Node failed	critical	
free_flash	Flash storage has reached % of its capacity	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.
insufficient_disk_aofrw	Node has insufficient disk space for AOF rewrite	true: error false: info	
memory	Node memory has reached % of its capacity	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.
net_throughput	Network throughput has reached MB/s	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.
persistent_storage	Persistent storage has reached % of its capacity	true: warning false: info	Has global_threshold parameter in the key/value section of the log entry.

Non-UI events

Logged events that do not appear in the UI

Event code name	Severity	Notes
checks_error	error	Indicates that one or more node checks have failed
node_abort_remove_request	info	
node_remove_request	info	

Updated: August 31, 2022

User event logs

The following user events can appear in syslog.

Non-UI events

Logged events that do not appear in the UI

Event code name	Severity	Notes
user_created	info	
user_deleted	info	
user_updated	info	Indicates that a user configuration has been updated

Updated: August 31, 2022

View Redis slow log

On the [Databases > Slowlog](#) page, you can view Slow Log details for Redis Enterprise Software databases.

[Redis Slow Log](#) is one of the best tools for debugging and tracing your Redis database, especially if you experience high latency and high CPU usage with Redis operations. Because Redis is based on a single threaded architecture, Redis Slow Log can be much more useful than slow log mechanisms of multi-threaded database systems such as MySQL Slow Query Log.

Unlike tools that introduce lock overhead (which complicates the debugging process), Redis Slow Log is highly effective at showing the actual processing time of each command.

Redis Enterprise Software includes enhancements to the standard Redis Slow Log capabilities that allow you to analyze the execution time complexity of each command. This enhancement can help you better analyze Redis operations, allowing you to compare the differences between execution times of the

same command, observe spikes in CPU usage, and more.

This is especially useful with complex commands such as [ZUNIONSTORE](#), [ZINTERSTORE](#) and [ZRANGEBYSCORE](#).

The enhanced Redis Enterprise Software Slow Log adds the **Complexity info** field to the output data.

View the complexity info data by its respective command in the table below:

Command	Value of interest	Complexity
LINSERT	N - list len	O(N)
LREM	N - list len	O(N)
LTRIM	N - number of removed elements	O(N)
PUBLISH	N - number of channel subscribers M - number of subscribed patterns	O(N+M)
PSUBSCRIBE	N - number of patterns client is subscribed to argc - number of arguments passed to the command	O(argc*N)
PUNSUBSCRIBE	N - number of patterns client is subscribed to M - total number of subscribed patterns argc - number of arguments passed to the command	O(argc*(N+M))
SDIFF	N - total number of elements in all sets	O(N)
SDIFFSTORE	N - total number of elements in all sets	O(N)
SINTER	N - number of elements in smallest set argc - number of arguments passed to the command	O(argc*N)
SINTERSTORE	N - number of elements in smallest set argc - number of arguments passed to the command	O(argc*N)
SMEMBERS	N - number of elements in a set	O(N)
SORT	N - number of elements in the when no sorting list/set/zset M - number of elements in result	O(N+M*log(M))O(N)
SUNION	N - number of elements in all sets	O(N)
SUNIONSTORE	N - number of elements in all sets	O(N)
UNSUBSCRIBE	N - total number of clients subscribed to all channels	O(N)
ZADD	N - number of elements in the zset	O(log(N))
ZCOUNT	N - number of elements in the zset M - number of elements between min and max	O(log(N)+M)
ZINCRBY	N - number of elements in the zset	O(log(N))
ZINTERSTORE	N - number of elements in the smallest zset K - number of zsets M - number of elements in the results set	O(N*K)+O(M*log(M))
ZRANGE	N - number of elements in the zset M - number of results	O(log(N)+M)
ZRANGEBYSCORE	N - number of elements in the zset M - number of results	O(log(N)+M)
ZRANK	N - number of elements in the zset	O(log(N))
ZREM	N - number of elements in the zset argc - number of arguments passed to the command	O(argc*log(N))
ZREMRANGEBYRANK	N - number of elements in the zset argc - number of arguments passed to the command	O(log(N)+M)
ZREMRANGEBYSCORE	N - number of elements in the zset M - number of elements removed	O(log(N)+M)
ZREVRANGE	N - number of elements in the zset M - number of results	O(log(N)+M)
ZREVRANK	N - number of elements in the zset	O(log(N))
ZUNIONSTORE	N - sum of element counts of all zsets M - element count of result	O(N)+O(M*log(M))

Monitoring with metrics and alerts

You can use the metrics that measure the performance of your Redis Enterprise Software (RS) clusters, nodes, databases and shards to keep an eye on the performance of your databases. In the RS admin console, you can see the real-time metrics and you can configure alerts that send notifications based on alert parameters.

You can also access the metrics and configure alerts through the REST API so that you can integrate the RS metrics into your monitoring environment, for example, using [Prometheus with Grafana](#) or [Uptrace](#).

Make sure you read the [definition of each metric](#) so that you understand exactly what it represents.

Real-time metrics

You can see the metrics of the cluster in:

- **Cluster > Metrics**
- **Node > Metrics** for each node
- **Database > Metrics** for each database, including the shards for that database

The scale selector at the top of the page allows you to set the X-axis (time) scale of the graph.

To choose which metrics to display in the two large graphs at the top of the page:

1. Hover over the graph you want to show in a large graph.
2. Click on the right or left arrow to choose which side to show the graph.

We recommend that you show two similar metrics in the top graphs so you can compare them side-by-side.

Cluster alerts

In **Cluster > Alert Settings**, you can enable alerts for node or cluster events, such as high memory usage or throughput.

Configured alerts are shown:

- As a notification on the status icon () for the node and cluster
- In the **log**
- In email notifications, if you configure [email alerts](#)



Note: If you enable alerts for “Node joined” or “Node removed” actions, you must also enable “Receive email alerts” so that the notifications are sent.

To enable alerts for a cluster:

1. In **Cluster > Alert Settings**, click **Edit**.
2. Select the alerts that you want to show for the cluster and click **Save**.

Database alerts

For each database, you can enable alerts for database events, such as high memory usage or throughput.

Configured alerts are shown:

- As a notification on the status icon () for the database
- In the **log**
- In emails, if you configure [email alerts](#)

To enable alerts for a database:

1. In **Configuration** for the database, click **Edit**.
2. Select the **Alerts** section to open it.
3. Select the alerts that you want to show for the database and click **Save**.

Send alerts by email

To send cluster and database alerts by email:

1. In Cluster > Alert Settings, click **Edit**.
2. Select **Set an email** to configure the [email server settings](#).
3. In Configuration for the database, click **Edit**.
4. Select the **Alerts** section to open it.
5. Select **Receive email alerts** and click **Save**.
6. In Access Control, select the [database and cluster alerts](#) that you want each user to receive.

Updated: August 17, 2023

Prometheus integration with Redis Enterprise Software

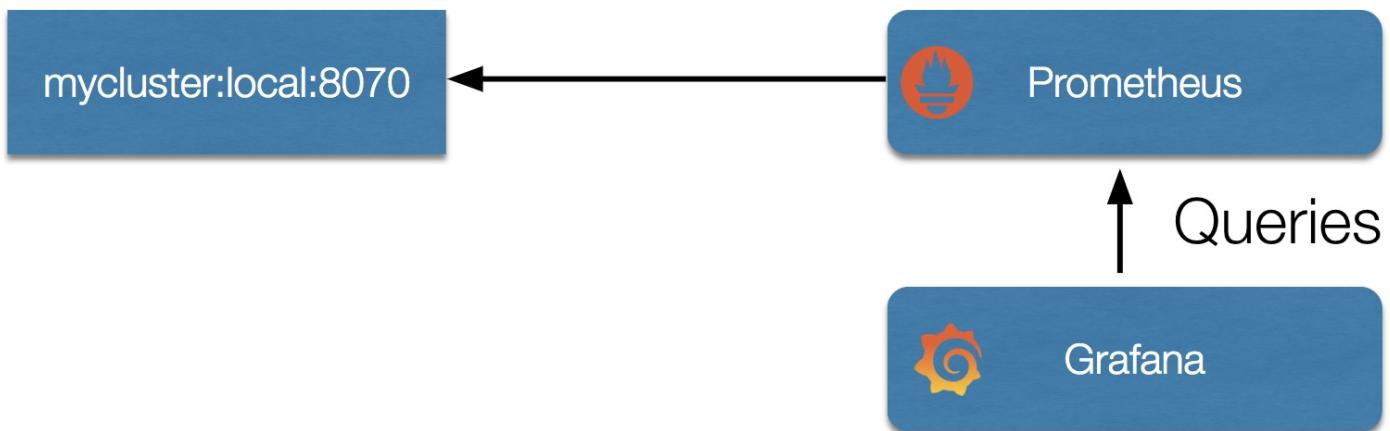
You can use Prometheus and Grafana to collect and visualize your Redis Enterprise Software metrics.

Metrics are exposed at the cluster, node, database, shard, and proxy levels.

- [Prometheus](#) is an open source systems monitoring and alerting toolkit that aggregates metrics from different sources.
- [Grafana](#) is an open source metrics visualization tool that processes Prometheus data.

You can use Prometheus and Grafana to:

- Collect and display metrics not available in the [admin console](#)
- Set up automatic alerts for node or cluster events
- Display Redis Enterprise Software metrics alongside data from other systems



In each cluster, the metrics_exporter process exposes Prometheus metrics on port 8070.

Quick start

To get started with Prometheus and Grafana:

1. Create a directory called 'prometheus' on your local machine.
2. Within that directory, create a configuration file called `prometheus.yml`.
3. Add the following contents to the configuration file and replace `<cluster_name>` with your Redis Enterprise cluster's FQDN:

 | Note: We recommend running Prometheus in Docker only for development and testing.

```

global:
  scrape_interval: 15s
  evaluation_interval: 15s

# Attach these labels to any time series or alerts when communicating with
# external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: "prometheus-stack-monitor"

# Load and evaluate rules in this file every 'evaluation_interval' seconds.
#rule_files:
# - "first.rules"
# - "second.rules"

scrape_configs:
# scrape Prometheus itself
- job_name: prometheus
  scrape_interval: 10s
  scrape_timeout: 5s
  static_configs:
    - targets: ["localhost:9090"]

# scrape Redis Enterprise
- job_name: redis-enterprise
  scrape_interval: 30s
  scrape_timeout: 30s
  metrics_path: /
  scheme: https
  tls_config:
    insecure_skip_verify: true
  static_configs:
    - targets: ["<cluster_name>:8070"]

```

4. Set up your Prometheus and Grafana servers. To set up Prometheus and Grafana on Docker:

1. Create a `docker-compose.yml` file:

```

version: '3'
services:
  prometheus-server:
    image: prom/prometheus
    ports:
      - 9090:9090
    volumes:
      - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml

  grafana-ui:
    image: grafana/grafana
    ports:
      - 3000:3000
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=secret
    links:
      - prometheus-server:prometheus

```

2. To start the containers, run:

```
$ docker compose up -d
```

3. To check that all of the containers are up, run: `docker ps`

4. In your browser, sign in to Prometheus at `http://localhost:9090` to make sure the server is running.

5. Select **Status** and then **Targets** to check that Prometheus is collecting data from your Redis Enterprise cluster.

redis-enterprise (1/1 up) show less					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://cluster.local:8070/	UP	<code>instance="cluster-local:8070"</code> <code>job="redis-enterprise"</code>	4.225s ago	169.676ms	

If Prometheus is connected to the cluster, you can type `node_up` in the Expression field on the Prometheus home page to see the cluster metrics.

5. Configure the Grafana datasource:

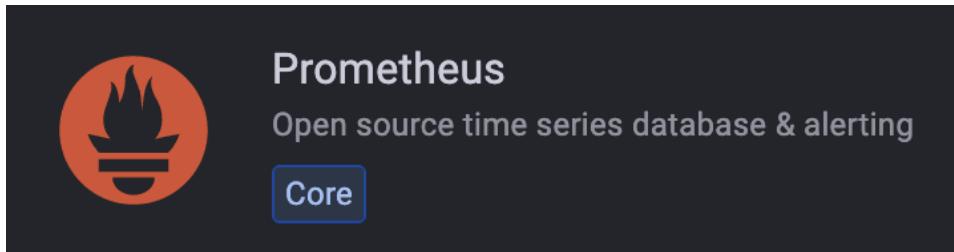
- Sign in to Grafana. If you installed Grafana locally, go to `http://localhost:3000` and sign in with:

- Username: admin
- Password: secret

- In the Grafana configuration menu, select **Data Sources**.

- Select **Add data source**.

- Select **Prometheus** from the list of data source types.



- Enter the Prometheus configuration information:

- Name: `redis-enterprise`
- URL: `http://<your prometheus address>:9090`
- Access: Server



Note:

- If the network port is not accessible to the Grafana server, select the **Browser** option from the Access menu.
- In a testing environment, you can select **Skip TLS verification**.

- Add dashboards for cluster, database, node, and shard metrics. To add preconfigured dashboards:

- In the Grafana dashboards menu, select **Manage**.
- Click **Import**.
- Upload one or more [Grafana dashboards](#).

Grafana dashboards for Redis Enterprise

Redis publishes four preconfigured dashboards for Redis Enterprise and Grafana:

- The [cluster status dashboard](#) provides an overview of your Redis Enterprise clusters.
- The [database status dashboard](#) displays specific database metrics, including latency, memory usage, ops/second, and key count.
- The [node metrics dashboard](#) provides metrics for each of the nodes hosting your cluster.
- The [shard metrics dashboard](#) displays metrics for the individual Redis processes running on your cluster nodes.

These dashboards are open source. For additional dashboard options, or to file an issue, see the [Redis Enterprise Grafana Dashboards Github repository](#).

For more information about configuring Grafana dashboards, see the [Grafana documentation](#).

Updated: April 19, 2023

Metrics in Prometheus

The [integration with Prometheus](#) lets you create dashboards that highlight the metrics that are important to you.

Here are the metrics available to Prometheus:

Database metrics

Metric	Description
bdb_avg_latency	Average latency of operations on the DB (seconds); returned only when there is traffic
bdb_avg_latency_max	Highest value of average latency of operations on the DB (seconds); returned only when there is traffic
bdb_avg_read_latency	Average latency of read operations (seconds); returned only when there is traffic
bdb_avg_read_latency_max	Highest value of average latency of read operations (seconds); returned only when there is traffic
bdb_avg_write_latency	Average latency of write operations (seconds); returned only when there is traffic
bdb_avg_write_latency_max	Highest value of average latency of write operations (seconds); returned only when there is traffic
bdb_bigstore_shard_count	Shard count by database and by storage engine (driver - rocksdb / speedb); Only for databases with Auto Tiering enabled
bdb_conn	Number of client connections to DB
bdb_egress_bytes	Rate of outgoing network traffic from the DB (bytes/sec)
bdb_egress_bytes_max	Highest value of rate of outgoing network traffic from the DB (bytes/sec)
bdb_evicted_objects	Rate of key evictions from DB (evictions/sec)
bdb_evicted_objects_max	Highest value of rate of key evictions from DB (evictions/sec)
bdb_expired_objects	Rate keys expired in DB (expirations/sec)
bdb_expired_objects_max	Highest value of rate keys expired in DB (expirations/sec)
bdb_fork_cpu_system	% cores utilization in system mode for all redis shard fork child processes of this database
bdb_fork_cpu_system_max	Highest value of % cores utilization in system mode for all redis shard fork child processes of this database
bdb_fork_cpu_user	% cores utilization in user mode for all redis shard fork child processes of this database
bdb_fork_cpu_user_max	Highest value of % cores utilization in user mode for all redis shard fork child processes of this database
bdb_ingress_bytes	Rate of incoming network traffic to DB (bytes/sec)
bdb_ingress_bytes_max	Highest value of rate of incoming network traffic to DB (bytes/sec)
bdb_instantaneous_ops_per_sec	Request rate handled by all shards of DB (ops/sec)

Metric	Description
bdb_main_thread_cpu_system	% cores utilization in system mode for all redis shard main threads of this database
bdb_main_thread_cpu_system_max	Highest value of % cores utilization in system mode for all redis shard main threads of this database
bdb_main_thread_cpu_user	% cores utilization in user mode for all redis shard main threads of this database
bdb_main_thread_cpu_user_max	Highest value of % cores utilization in user mode for all redis shard main threads of this database
bdb_mem_frag_ratio	RAM fragmentation ratio (RSS / allocated RAM)
bdb_mem_size_lua	Redis lua scripting heap size (bytes)
bdb_memory_limit	Configured RAM limit for the database
bdb_monitor_sessions_count	Number of client connected in monitor mode to the DB
bdb_no_of_keys	Number of keys in DB
bdb_other_req	Rate of other (non read/write) requests on DB (ops/sec)
bdb_other_req_max	Highest value of rate of other (non read/write) requests on DB (ops/sec)
bdb_other_res	Rate of other (non read/write) responses on DB (ops/sec)
bdb_other_res_max	Highest value of rate of other (non read/write) responses on DB (ops/sec)
bdb_pubsub_channels	Count the pub/sub channels with subscribed clients
bdb_pubsub_channels_max	Highest value of count the pub/sub channels with subscribed clients
bdb_pubsub_patterns	Count the pub/sub patterns with subscribed clients
bdb_pubsub_patterns_max	Highest value of count the pub/sub patterns with subscribed clients
bdb_read_hits	Rate of read operations accessing an existing key (ops/sec)
bdb_read_hits_max	Highest value of rate of read operations accessing an existing key (ops/sec)
bdb_read_misses	Rate of read operations accessing a non-existing key (ops/sec)
bdb_read_misses_max	Highest value of rate of read operations accessing a non-existing key (ops/sec)
bdb_read_req	Rate of read requests on DB (ops/sec)
bdb_read_req_max	Highest value of rate of read requests on DB (ops/sec)
bdb_read_res	Rate of read responses on DB (ops/sec)
bdb_read_res_max	Highest value of rate of read responses on DB (ops/sec)
bdb_shard_cpu_system	% cores utilization in system mode for all redis shard processes of this database
bdb_shard_cpu_system_max	Highest value of % cores utilization in system mode for all redis shard processes of this database
bdb_shard_cpu_user	% cores utilization in user mode for the redis shard process
bdb_shard_cpu_user_max	Highest value of % cores utilization in user mode for the redis shard process
bdb_shards_used	Used shard count by database and by shard type (ram / flash)
bdb_total_connections_received	Rate of new client connections to DB (connections/sec)
bdb_total_connections_received_max	Highest value of rate of new client connections to DB (connections/sec)
bdb_total_req	Rate of all requests on DB (ops/sec)
bdb_total_req_max	Highest value of rate of all requests on DB (ops/sec)
bdb_total_res	Rate of all responses on DB (ops/sec)
bdb_total_res_max	Highest value of rate of all responses on DB (ops/sec)
bdb_up	Database is up and running
bdb_used_memory	Memory used by db (in bigredis this includes flash) (bytes)
bdb_write_hits	Rate of write operations accessing an existing key (ops/sec)
bdb_write_hits_max	Highest value of rate of write operations accessing an existing key (ops/sec)
bdb_write_misses	Rate of write operations accessing a non-existing key (ops/sec)
bdb_write_misses_max	Highest value of rate of write operations accessing a non-existing key (ops/sec)
bdb_write_req	Rate of write requests on DB (ops/sec)

Metric	Description
bdb_write_req_max	Highest value of rate of write requests on DB (ops/sec)
bdb_write_res	Rate of write responses on DB (ops/sec)
bdb_write_res_max	Highest value of rate of write responses on DB (ops/sec)
no_of_expires	Current number of volatile keys in the database

Node metrics

Metric	Description
node_available_flash	Available flash in node (bytes)
node_available_flash_no_overbooking	Available flash in node (bytes), without taking into account overbooking
node_available_memory	Amount of free memory in node (bytes) that is available for database provisioning
node_available_memory_no_overbooking	Available ram in node (bytes) without taking into account overbooking
node_avg_latency	Average latency of requests handled by endpoints on node (seconds); returned only when there is traffic
node_bigstore_free	Sum of free space of back-end flash (used by flash DB's [BigRedis]) on all cluster nodes (bytes); returned only when BigRedis is enabled
node_bigstore_iops	Rate of i/o operations against back-end flash for all shards which are part of a flash based DB (BigRedis) in cluster (ops/sec); returned only when BigRedis is enabled
node_bigstore_kv_ops	Rate of value read/write operations against back-end flash for all shards which are part of a flash based DB (BigRedis) in cluster (ops/sec); returned only when BigRedis is enabled
node_bigstore_throughput	Throughput i/o operations against back-end flash for all shards which are part of a flash based DB (BigRedis) in cluster (bytes/sec); returned only when BigRedis is enabled
node_connss	Number of clients connected to endpoints on node
node_cpu_idle	CPU idle time portion (0-1, multiply by 100 to get percent)
node_cpu_idle_max	Highest value of CPU idle time portion (0-1, multiply by 100 to get percent)
node_cpu_idle_median	Average value of CPU idle time portion (0-1, multiply by 100 to get percent)
node_cpu_idle_min	Lowest value of CPU idle time portion (0-1, multiply by 100 to get percent)
node_cpu_system	CPU time portion spent in kernel (0-1, multiply by 100 to get percent)
node_cpu_system_max	Highest value of CPU time portion spent in kernel (0-1, multiply by 100 to get percent)
node_cpu_system_median	Average value of CPU time portion spent in kernel (0-1, multiply by 100 to get percent)
node_cpu_system_min	Lowest value of CPU time portion spent in kernel (0-1, multiply by 100 to get percent)
node_cpu_user	CPU time portion spent by users-space processes (0-1, multiply by 100 to get percent)
node_cpu_user_max	Highest value of CPU time portion spent by users-space processes (0-1, multiply by 100 to get percent)
node_cpu_user_median	Average value of CPU time portion spent by users-space processes (0-1, multiply by 100 to get percent)
node_cpu_user_min	Lowest value of CPU time portion spent by users-space processes (0-1, multiply by 100 to get percent)
node_cur_aof_rewrites	Number of aof rewrites that are currently performed by shards on this node
node_egress_bytes	Rate of outgoing network traffic to node (bytes/sec)
node_egress_bytes_max	Highest value of rate of outgoing network traffic to node (bytes/sec)
node_egress_bytes_median	Average value of rate of outgoing network traffic to node (bytes/sec)
node_egress_bytes_min	Lowest value of rate of outgoing network traffic to node (bytes/sec)
node_ephemeral_storage_avail	Disk space available to RLEC processes on configured ephemeral disk (bytes)
node_ephemeral_storage_free	Free disk space on configured ephemeral disk (bytes)

Metric	Description
node_free_memory	Free memory in node (bytes)
node_ingress_bytes	Rate of incoming network traffic to node (bytes/sec)
node_ingress_bytes_max	Highest value of rate of incoming network traffic to node (bytes/sec)
node_ingress_bytes_median	Average value of rate of incoming network traffic to node (bytes/sec)
node_ingress_bytes_min	Lowest value of rate of incoming network traffic to node (bytes/sec)
node_persistent_storage_avail	Disk space available to RLEC processes on configured persistent disk (bytes)
node_persistent_storage_free	Free disk space on configured persistent disk (bytes)
node_provisional_flash	Amount of flash available for new shards on this node, taking into account overbooking, max redis servers, reserved flash and provision and migration thresholds (bytes)
node_provisional_flash_no_overbooking	Amount of flash available for new shards on this node, without taking into account overbooking, max redis servers, reserved flash and provision and migration thresholds (bytes)
node_provisional_memory	Amount of RAM that is available for provisioning to databases out of the total RAM allocated for databases
node_provisional_memory_no_overbooking	Amount of RAM that is available for provisioning to databases out of the total RAM allocated for databases, without taking into account overbooking
node_total_req	Request rate handled by endpoints on node (ops/sec)
node_up	Node is part of the cluster and is connected

Cluster metrics

Metric	Description
cluster_shards_limit	Total shard limit by the license by shard type (ram / flash)

Proxy metrics

Metric	Description
listener_acc_latency	Accumulative latency (sum of the latencies) of all types of commands on DB. For the average latency, divide this value by listener_total_res
listener_acc_latency_max	Highest value of accumulative latency of all types of commands on DB
listener_acc_other_latency	Accumulative latency (sum of the latencies) of commands that are type "other" on DB. For the average latency, divide this value by listener_other_res
listener_acc_other_latency_max	Highest value of accumulative latency of commands that are type "other" on DB
listener_acc_read_latency	Accumulative latency (sum of the latencies) of commands that are type "read" on DB. For the average latency, divide this value by listener_read_res
listener_acc_read_latency_max	Highest value of accumulative latency of commands that are type "read" on DB
listener_acc_write_latency	Accumulative latency (sum of the latencies) of commands that are type "write" on DB. For the average latency, divide this value by listener_write_res
listener_acc_write_latency_max	Highest value of accumulative latency of commands that are type "write" on DB
listener_auth_cmds	Number of memcached AUTH commands sent to the DB
listener_auth_cmds_max	Highest value of number of memcached AUTH commands sent to the DB
listener_auth_errors	Number of error responses to memcached AUTH commands
listener_auth_errors_max	Highest value of number of error responses to memcached AUTH commands
listener_cmd_flush	Number of memcached FLUSH_ALL commands sent to the DB
listener_cmd_flush_max	Highest value of number of memcached FLUSH_ALL commands sent to the DB
listener_cmd_get	Number of memcached GET commands sent to the DB

Metric	Description
listener_cmd_get_max	Highest value of number of memcached GET commands sent to the DB
listener_cmd_set	Number of memcached SET commands sent to the DB
listener_cmd_set_max	Highest value of number of memcached SET commands sent to the DB
listener_cmd_touch	Number of memcached TOUCH commands sent to the DB
listener_cmd_touch_max	Highest value of number of memcached TOUCH commands sent to the DB
listener_conns	Number of clients connected to the endpoint
listener_egress_bytes	Rate of outgoing network traffic to the endpoint (bytes/sec)
listener_egress_bytes_max	Highest value of rate of outgoing network traffic to the endpoint (bytes/sec)
listener_ingress_bytes	Rate of incoming network traffic to the endpoint (bytes/sec)
listener_ingress_bytes_max	Highest value of rate of incoming network traffic to the endpoint (bytes/sec)
listener_last_req_time	Time of last command sent to the DB
listener_last_res_time	Time of last response sent from the DB
listener_max_connections_exceeded	Number of times the Number of clients connected to the db at the same time has exceeded the max limit
listener_max_connections_exceeded_max	Highest value of number of times the Number of clients connected to the db at the same time has exceeded the max limit
listener_monitor_sessions_count	Number of client connected in monitor mode to the endpoint
listener_other_req	Rate of other (non read/write) requests on the endpoint (ops/sec)
listener_other_req_max	Highest value of rate of other (non read/write) requests on the endpoint (ops/sec)
listener_other_res	Rate of other (non read/write) responses on the endpoint (ops/sec)
listener_other_res_max	Highest value of rate of other (non read/write) responses on the endpoint (ops/sec)
listener_other_started_res	Number of responses sent from the DB of type "other"
listener_other_started_res_max	Highest value of number of responses sent from the DB of type "other"
listener_read_req	Rate of read requests on the endpoint (ops/sec)
listener_read_req_max	Highest value of rate of read requests on the endpoint (ops/sec)
listener_read_res	Rate of read responses on the endpoint (ops/sec)
listener_read_res_max	Highest value of rate of read responses on the endpoint (ops/sec)
listener_read_started_res	Number of responses sent from the DB of type "read"
listener_read_started_res_max	Highest value of number of responses sent from the DB of type "read"
listener_total_connections_received	Rate of new client connections to the endpoint (connections/sec)
listener_total_connections_received_max	Highest value of rate of new client connections to the endpoint (connections/sec)
listener_total_req	Request rate handled by the endpoint (ops/sec)
listener_total_req_max	Highest value of rate of all requests on the endpoint (ops/sec)
listener_total_res	Rate of all responses on the endpoint (ops/sec)
listener_total_res_max	Highest value of rate of all responses on the endpoint (ops/sec)
listener_total_started_res	Number of responses sent from the DB of all types
listener_total_started_res_max	Highest value of number of responses sent from the DB of all types
listener_write_req	Rate of write requests on the endpoint (ops/sec)
listener_write_req_max	Highest value of rate of write requests on the endpoint (ops/sec)
listener_write_res	Rate of write responses on the endpoint (ops/sec)
listener_write_res_max	Highest value of rate of write responses on the endpoint (ops/sec)
listener_write_started_res	Number of responses sent from the DB of type "write"
listener_write_started_res_max	Highest value of number of responses sent from the DB of type "write"

Replication metrics

Metric	Description
bdb_replicaof_syncer_ingress_bytes	Rate of compressed incoming network traffic to Replica Of DB (bytes/sec)
bdb_replicaof_syncer_ingress_bytes_decompressed	Rate of decompressed incoming network traffic to Replica Of DB (bytes/sec)
bdb_replicaof_syncer_local_ingress_lag_time	Lag time between the source and the destination for Replica Of traffic (ms)
bdb_replicaof_syncer_status	Syncer status for Replica Of traffic; 0 = in-sync, 1 = syncing, 2 = out of sync
bdb_crdt_syncer_ingress_bytes	Rate of compressed incoming network traffic to CRDB (bytes/sec)
bdb_crdt_syncer_ingress_bytes_decompressed	Rate of decompressed incoming network traffic to CRDB (bytes/sec)
bdb_crdt_syncer_local_ingress_lag_time	Lag time between the source and the destination (ms) for CRDB traffic
bdb_crdt_syncer_status	Syncer status for CRDB traffic; 0 = in-sync, 1 = syncing, 2 = out of sync

Shard metrics

Metric	Description
redis_active_defrag_running	Automatic memory defragmentation current aggressiveness (% cpu)
redis_allocator_active	Total used memory including external fragmentation
redis_allocator_allocated	Total allocated memory
redis_allocator_resident	Total resident memory (RSS)
redis_aof_last_cow_size	Last AOFR, CopyOnWrite memory
redis_aof_rewrite_in_progress	The number of simultaneous AOF rewrites that are in progress
redis_aof_rewrites	Number of AOF rewrites this process executed
redis_aof_delayed_fsync	Number of times an AOF fsync caused delays in the redis main thread (inducing latency); This can indicate that the disk is slow or overloaded
redis_blocked_clients	Count the clients waiting on a blocking call
redis_connected_clients	Number of client connections to the specific shard
redis_connected_slaves	Number of connected slaves
redis_db0_avg_ttl	Average TTL of all volatile keys
redis_db0_expires	Total count of volatile keys
redis_db0_keys	Total key count
redis_evicted_keys	Keys evicted so far (since restart)
redis_expire_cycle_cpu_milliseconds	The cumulative amount of time spent on active expiry cycles
redis_expired_keys	Keys expired so far (since restart)
redis_forwarding_state	Shard forwarding state (on or off)
redis_keys_trimmed	The number of keys that were trimmed in the current or last resharding process
redis_keyspace_read_hits	Number of read operations accessing an existing keyspace
redis_keyspace_read_misses	Number of read operations accessing an non-existing keyspace
redis_keyspace_write_hits	Number of write operations accessing an existing keyspace
redis_keyspace_write_misses	Number of write operations accessing an non-existing keyspace
redis_master_link_status	Indicates if the replica is connected to its master
redis_master_repl_offset	Number of bytes sent to replicas by the shard; Calculate the throughput for a time period by comparing the value at different times
redis_master_sync_in_progress	The master shard is synchronizing (1 true)
redis_max_process_mem	Current memory limit configured by redis_mgr according to node free memory
redis_maxmemory	Current memory limit configured by redis_mgr according to db memory limits
redis_mem_aof_buffer	Current size of AOF buffer
redis_mem_clients_normal	Current memory used for input and output buffers of non-replica clients
redis_mem_clients_slaves	Current memory used for input and output buffers of replica clients
redis_mem_fragmentation_ratio	Memory fragmentation ratio (1.3 means 30% overhead)

Metric	Description
redis_mem_not_counted_for_evict	Portion of used_memory (in bytes) that's not counted for eviction and OOM error
redis_mem_replication_backlog	Size of replication backlog
redis_module_fork_in_progress	A binary value that indicates if there is an active fork spawned by a module (1) or not (0)
redis_process_cpu_system_seconds_total	Shard Process system CPU time spent in seconds
redis_process_cpu_usage_percent	Shard Process cpu usage precentage
redis_process_cpu_user_seconds_total	Shard user CPU time spent in seconds
redis_process_main_thread_cpu_system_seconds_total	Shard main thread system CPU time spent in seconds
redis_process_main_thread_cpu_user_seconds_total	Shard main thread user CPU time spent in seconds
redis_process_max_fds	Shard Maximum number of open file descriptors
redis_process_open_fds	Shard Number of open file descriptors
redis_process_resident_memory_bytes	Shard Resident memory size in bytes
redis_process_start_time_seconds	Shard Start time of the process since unix epoch in seconds
redis_process_virtual_memory_bytes	Shard virtual memory in bytes
redis_rdb_bgsave_in_progress	Indication if bgsave is currently in progress
redis_rdb_last_cow_size	Last bgsave (or SYNC fork) used CopyOnWrite memory
redis_rdb_saves	Total count of bgsaves since process was restarted (including replica fullsync and persistence)
redis_repl_touch_bytes	Number of bytes sent to replicas as TOUCH commands by the shard as a result of a READ command that was processed; Calculate the throughput for a time period by comparing the value at different times
redis_total_commands_processed	Number of commands processed by the shard; Calculate the number of commands for a time period by comparing the value at different times
redis_total_connections_received	Number of connections received by the shard; Calculate the number of connections for a time period by comparing the value at different times
redis_total_net_input_bytes	Number of bytes received by the shard; Calculate the throughput for a time period by comparing the value at different times
redis_total_net_output_bytes	Number of bytes sent by the shard; Calculate the throughput for a time period by comparing the value at different times
redis_up	Shard is up and running
redis_used_memory	Memory used by shard (in bigredis this includes flash) (bytes)

Updated: August 17, 2023

Uptrace integration with Redis Enterprise Software

To collect, view, and monitor metrics data from your databases and other cluster components, you can connect Uptrace to your Redis Enterprise cluster using OpenTelemetry Collector.

Uptrace is an [open source APM tool](#) that supports distributed tracing, metrics, and logs. You can use it to monitor applications and set up automatic alerts to receive notifications.

Uptrace uses OpenTelemetry to collect and export telemetry data from software applications such as Redis. OpenTelemetry is an open source observability framework that aims to provide a single standard for all types of observability signals such as traces, metrics, and logs.

With OpenTelemetry Collector, you can receive, process, and export telemetry data to any [OpenTelemetry backend](#). You can also use Collector to scrape Prometheus metrics provided by Redis and then export those metrics to Uptrace.

You can use Uptrace to:

- Collect and display data metrics not available in the [admin console](#).
- Use prebuilt dashboard templates maintained by the Uptrace community.
- Set up automatic alerts and receive notifications via email, Slack, Telegram, and others.
- Monitor your app performance and logs using [OpenTelemetry tracing](#).



DASHBOARDS EXPLORE

Redis: Nodes ▾



CLONE



1 HOUR ▾



RELOAD

Nodes up

DBs up

Shards up

CPU usage

Used RAM

Avail. RAM

Requests rate

2 of 2

2 of 2

3 of 3

49%

2.3MB

2.8GB

15.1k/sec

Group by Where Aggregate

Edit Help

group by cluster X group by node X 1 - \$cpu_idle as cpu_usage X \$mem_available X \$conns X \$requests X \$ingress X \$egress X +

Redis: Nodes

EDIT

cluster	node	cpu_usage ↓	mem_available	conns	requests	ingress	egress
cluster1.local	1						
cluster2.local	1						

Install Collector and Uptrace

Because installing OpenTelemetry Collector and Uptrace can take some time, you can use the [docker-compose](#) example that also comes with Redis Enterprise cluster.

After you download the Docker example, you can edit the following configuration files in the `uptrace/example/redis-enterprise` directory before you start the Docker containers:

- `otel-collector.yaml` - Configures `/etc/otelcol-contrib/config.yaml` in the OpenTelemetry Collector container.
- `uptrace.yml` - Configures `/etc/uptrace/uptrace.yml` in the Uptrace container.

You can also install OpenTelemetry and Uptrace from scratch using the following guides:

- [Getting started with OpenTelemetry Collector](#)
- [Getting started with Uptrace](#)

After you install Uptrace, you can access the Uptrace UI at <http://localhost:14318/>.

Scrape Prometheus metrics

Redis Enterprise cluster exposes a Prometheus scraping endpoint on `http://localhost:8070/`. You can scrape that endpoint by adding the following lines to the OpenTelemetry Collector config:

```
# /etc/otelcol-contrib/config.yaml

prometheus_simple/cluster1:
  collection_interval: 10s
  endpoint: "localhost:8070" # Redis Cluster endpoint
  metrics_path: "/"
  tls:
    insecure: false
    insecure_skip_verify: true
    min_version: "1.0"
```

Next, you can export the collected metrics to Uptrace using OpenTelemetry protocol (OTLP):

```
# /etc/otelcol-contrib/config.yaml

receivers:
  otlp:
    protocols:
      grpc:
      http:

exporters:
  otlp/uptrace:
    # Uptrace is accepting metrics on this port
    endpoint: localhost:14317
    headers: { "uptrace-dsn": "http://project1_secret_token@localhost:14317/1" }
    tls: { insecure: true }

service:
  pipelines:
    traces:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp/uptrace]
    metrics:
      receivers: [otlp, prometheus_simple/cluster1]
      processors: [batch]
      exporters: [otlp/uptrace]
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [otlp/uptrace]
```

Don't forget to restart the Collector and then check logs for any errors:

```
docker-compose logs otel-collector

# or

sudo journalctl -u otelcol-contrib -f
```

You can also check the full OpenTelemetry Collector config[here](#).

View metrics

When metrics start arriving to Uptrace, you should see a couple of dashboards in the Metrics tab. In total, Uptrace should create 3 dashboards for Redis Enterprise metrics:

- “Redis: Nodes” dashboard displays a list of cluster nodes. You can select a node to view its metrics.
- “Redis: Databases” displays a list of Redis databases in all cluster nodes. To find a specific database, you can use filters or sort the table by columns.
- “Redis: Shards” contains a list of shards that you have in all cluster nodes. You can filter or sort shards and select a shard for more details.

Monitor metrics

To start monitoring metrics, you need to create metrics monitors using Uptrace UI:

- Open “Alerts” -> “Monitors”.
- Click “Create monitor” -> “Create metrics monitor”.

For example, the following monitor uses the `group by node` expression to create an alert whenever an individual Redis shard is down:

```

monitors:
  - name: Redis shard is down
    metrics:
      - redis_up as $redis_up
    query:
      - group by cluster # monitor each cluster,
      - group by bdb # each database,
      - group by node # and each shard
      - $redis_up
    min_allowed_value: 1
    # shard should be down for 5 minutes to trigger an alert
    for_duration: 5m

```

You can also create queries with more complex expressions.

For example, the following monitors create an alert when the keyspace hit rate is lower than 75% or memory fragmentation is too high:

```

monitors:
  - name: Redis read hit rate < 75%
    metrics:
      - redis_keyspace_read_hits as $hits
      - redis_keyspace_read_misses as $misses
    query:
      - group by cluster
      - group by bdb
      - group by node
      - $hits / ($hits + $misses) as hit_rate
    min_allowed_value: 0.75
    for_duration: 5m

  - name: Memory fragmentation is too high
    metrics:
      - redis_used_memory as $mem_used
      - redis_mem_fragmentation_ratio as $fragmentation
    query:
      - group by cluster
      - group by bdb
      - group by node
      - where $mem_used > 32mb
      - $fragmentation
    max_allowed_value: 3
    for_duration: 5m

```

You can learn more about the query language [here](#).

Updated: August 17, 2023

Nagios integration with Redis Enterprise Software

The Redis Enterprise Software (RS) Nagios plugin enables you to monitor the status of RS related objects and alerts. The RS alerts can be related to the cluster, nodes, or databases.

The alerts that can be monitored via Nagios are the same alerts that can be configured in the RS UI in the Settings > Alerts page, or the specific Database > Configuration page.

All alert configurations (active / not active, setting thresholds, etc') can only be done through the RS UI, they cannot be configured in Nagios. Through Nagios you can only view the status and information of the alerts.

The full list of alerts can be found in the plugin package itself (in "/rlec_obj/rlec_services.cfg" file, more details below).

RS Nagios plugin support API password retrieval from Gnome keyring, KWallet, Windows credential vault, Mac OS X Keychain, if present, or otherwise Linux Secret Service compatible password store. With no keyring service available, the password is saved with base64 encoding, under the user home directory.

Configuring the Nagios plugin

In order to configure the Nagios plugin you need to copy the files that come with the package into your Nagios environment and place them in a Nagios configuration directory. Or, alternatively you can copy parts of the package configuration into your existing Nagios configuration.

If Keyring capabilities are needed to store the password, python keyring package should be installed and used by following the below steps from the operating system CLI on Nagios machine:

1. pip install keyring to install the package (See <https://pip.pypa.io/en/stable/installing/> on how to install python pip if needed).
2. keyring set RS-Nagios <RS user email> to set the password. User email should be identical to the email used in Nagios configuration and the password should be set using the same user that run the Nagios server.

Then, you need to update the local parameters, such as hostnames, addresses, and object IDs, to the values relevant for your RS deployment.

Finally, you need to set the configuration for each node and database you would like to monitor. More details below.

The RS Nagios package includes two components:

- The plugin itself - with suffix "rlec_nagios_plugin"
- Configuration files - with suffix "rlec_nagios_conf"

Below is the list of files included in these packages and instructions regarding what updates need to be made to these files.

Note : The instructions below assume you are running on Ubuntu, have a clean Nagios installation, and the base Nagios directory is "/usr/local/nagios/"

Step 1

Copy the folder named "libexec" from the plugin folder and its contents to "/usr/local/nagios/"

These files included in it are:

- check_rlec_alert
- check_rlec_node
- check_rlec_bdb
- email_stub
- rlecdigest.py

Note : The check_rlec_alert, check_rlec_node, check_rlec_bdb files are the actual plugin implementation. You can run each of them with a "h" switch in order to retrieve their documentation and their expected parameters.

Step 2

Add the following lines to your "nagios.cfg":

- cfg_dir=/usr/local/nagios/etc/rlec_obj
- cfg_dir=/usr/local/nagios/etc/rlec_local
- resource_file=/usr/local/nagios/etc/rlec_resource.cfg

Step 3

Copy the configuration files along with their folders to "/usr/local/nagios/etc" and make the required updates, as detailed below.

1. Under the "/etc" folder:
 1. "rlec_resource.cfg" holds global variables definitions for the user and password to use to connect to RS. You should update the variables to the relevant user and password for your deployment.
 2. "rlec_local" folder
 3. "rlec_obj" folder
2. Under the "/rlec_local" folder:
 1. "cluster.cfg" holds configuration details at the cluster level. If you would like to monitor more than one cluster then you need to duplicate the two existing entries in the file for each cluster.
 1. The first "define host" section defines a variable for the IP address of the cluster that is used in other configuration files.
 1. Update the "address" to the Cluster Name (FQDN) as defined in DNS, or the IP address of one of the nodes in the cluster.
 2. If you are configuring more than one RS then when duplicating this section you should make sure:
 1. The "name" is unique.
 2. In the second "define host" section:
 1. The "host_name" in each entry must be unique.
 2. The "display_name" in each entry can be updated to a user-friendly name that are shown in Nagios UI.
 2. "contacts.cfg" holds configuration details who to send emails to. It should be updated to values relevant for your deployment. If this file already exists in your existing Nagios environment then you should update it accordingly.
 3. "databases.cfg" holds configuration details of the databases to monitor. The "define host" section should be duplicated for every database to

monitor.

1. "host_name" should be a unique value.
2. "display_name" should be updated to a user-friendly name to show in the UI.
3. "_RLECID" should be the database's internal ID that can be retrieved from `radmin status` command output.
4. "nodes.cfg" holds configuration details of the nodes in the cluster. The "define host" section should be duplicated for every node in the cluster.
 1. "host_name" should be a unique value.
 2. "display_name" should be updated to a user-friendly name to show in the UI.
 3. "address" should be updated to the DNS name mapped to the IP address of the node, or to the IP address itself.
 4. "_RLECID" should be the node's internal ID that can be retrieved from `radmin status` command output.
5. Under the "/rlec_obj" folder:
 1. "rlec_cmd.cfg" holds configuration details of how to activate the plugin. No need to make any updates to it.
 2. "rlec_groups.cfg" holds definitions of host groups. No need to make any updates to it.
 3. "rlec_services.cfg" holds definitions of all alerts that are monitored. No need to make any updates to it.
 4. "rlec_templates.cfg" holds general RS Nagios definitions. No need to make any updates to it.

Updated: August 31, 2022

Manage networks

When designing a Redis Enterprise Software solution, there are some specific networking features that are worth your time to understand and implement.

Configure cluster DNS

Configure DNS to communicate between cluster nodes.

- [AWS Route53 DNS management](#)
- [Client prerequisites for mDNS](#) for development and test environments

Cluster load balancer setup

Set up a load balancer to direct traffic to cluster nodes when DNS is not available.

Multi-IP and IPv6

Requirements for using multiple IP addresses or IPv6 addresses with Redis Enterprise Software.

Network port configurations

Describes the port ranges that Redis Enterprise Software uses.

Public and private endpoints

Enable public and private endpoints for your databases.

Updated: February 10, 2023

AWS Route53 DNS management

Redis Enterprise Software requires DNS to be properly configured to achieve high availability and fail-over regardless of where it is installed.

Here, you learn how to configure AWS Route53 DNS resolution.

Prerequisites

You need to have a domain name registered. Then, either you need to have Amazon's Route53 as the primary/master nameserver (NS) for this domain or for a delegated zone under this domain. Finally, you need to have the zone (either the whole domain or a sub-zone) defined in AWS Route53.

How does Redis Enterprise Software achieve failover?

When your application wants to connect to a RS database in a three node Redis Enterprise Software cluster, it connects to any node in this cluster using its fully qualified domain name (FQDN), for example “node1.mycluster.enterprise.com”. It needs to know the IP address associated with this name and asks the top-level nameservers (.com) for the list of name servers in charge of enterprise.com. Then, it asks these name server (one after each other in case of failure) for the name servers in charge of mycluster.enterprise.com, and finally, it asks these name servers (one after each other in case of failure) for the IP of node1.mycluster.enterprise.com. At the end, it connects to this IP address. All this process is obfuscated from the application and is done by the resolver, a system library.

Your name servers are in charge of enterprise.com. So, they are able to give the name servers in charge of your cluster. RS embeds a name server in each node. The nodes are in charge of the cluster zone resolution and are able to give the IP address of any node in the cluster. When everything is working, whichever is the top-level nameserver asked, it gives the list of name servers for your enterprise domain, then whichever enterprise name server is asked, it gives the list of cluster name servers (the cluster nodes), and whichever is the node asked, it returns the IP address of the requested node name.

If the cluster nameserver (node) asked is down, given the resolution process, the resolver tries to ask another name server (node) from the list and gets the requested IP address. That's the standard DNS resolution behavior, in few words.

Now, when a cluster node dies, for whatever reason, the other nodes can not reach it anymore and replaces the IP address associated with his name by the IP address of another node in the cluster's name servers. It means that the failed over IP address is never returned anymore by the name servers to any requested FQDN, and that two of the FQDN have the same IP address. Basically, it means that whichever node FQDN your application is asking to resolve, it always gets the IP address of one node in the cluster that is healthy, up and running and the connection succeeds.

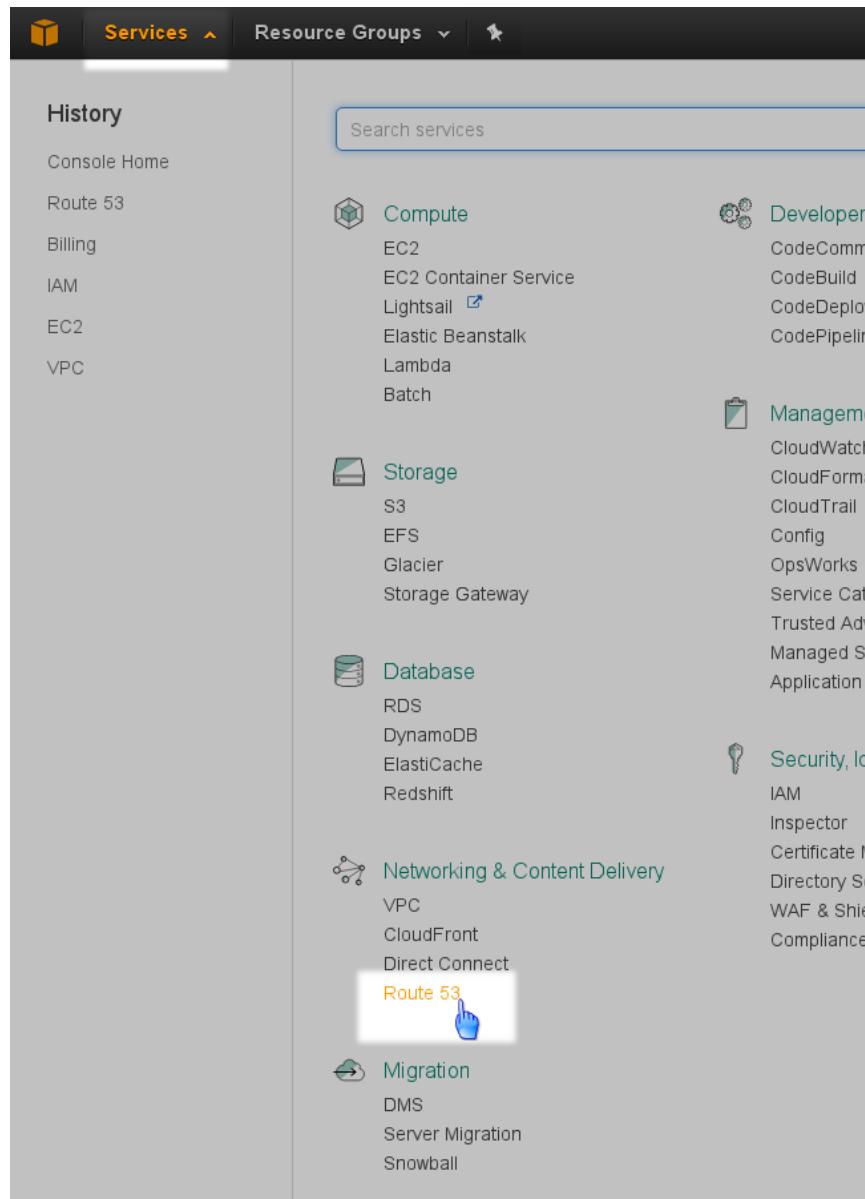
Configuration

After the theory, we can go through the hands-on steps to achieve this configuration with AWS's Route53 DNS as your domain or sub-domain official name server.

If you would like to watch a video on the process, here you go.

Connection to AWS Route53

The first step is to connect your browser to AWS and to login into the administration interface. Then, you have to go in the *Services* menu at the top of the page and click on the *Route53* menu item:



Make sure you have registered a domain, and that you have defined *Route53* as the primary/master name server for the whole domain or for one of its sub-domains. You should have at least one zone in *Route53*. Click on the *Hosted zones* link to open it:



Route53 is now displaying the list of the zones that you defined. You need to click on the zone in which you want to define your cluster, *demo-rlec.redislabs.com* in my case:

The screenshot shows the AWS Route 53 service dashboard. On the left sidebar, under the 'Hosted zones' section, the domain 'demo-rlc.redislabs.com.' is listed as public. At the top center, there are buttons for 'Create Hosted Zone', 'Go to Record Sets', and 'Delete Hosted Zone'. A search bar at the top allows searching across all fields.

Nameserver records creation

The next step is to create the record that returns the IP address of the name servers in your cluster, that is one of the nodes. To create the first name server IP address resolution record, you need to click on the *Create Record Set* blue button at the top of the list:

The screenshot shows the 'Create Record Set' dialog box overlaid on the main Route 53 interface. The dialog has a search bar for 'Record Set Name' and dropdowns for 'Type' (set to 'Any Type') and 'Value'. It also includes checkboxes for 'Aliases Only' and 'Weighted Only'. Below the dialog, a table lists existing records for the domain 'demo-rlc.redislabs.com.'. One record is selected, and a hand cursor is pointing at the 'Create Record Set' button in the dialog.

Name	Type	Value
demo-rlc.redislabs.com.	NS	ns-369.awsdns-46.com. ns-565.awsdns-06.net. ns-1983.awsdns-55.co.uk. ns-1183.awsdns-19.org.
demo-rlc.redislabs.com.	SOA	ns-369.awsdns-46.com. awsdns-hostmaster.amazon.

This record is **only** used to resolve the IP address of the cluster name server to query, it is **not** used by the application to connect to the cluster. This kind of record is an A record type and associates an IP address to a name. To avoid the whole resolving process each time that a name is requested, the results are cached in the forwarding DNS and in the application's resolver library. Given that the IP address associated with a node can change when the related hardware fails, the information needs to expire quickly, otherwise, the node fails, the name servers reflects the failover, but they are not queried again and the local resolver still returns the IP of the failed node. This is the Time To Live (TTL) field associated to the record.

So, you need to enter the name used as the name server's name, despite that it could be different, I suggest that you use the node name. You need to enter its IP address, to set the TTL to something short, I'll set it to one minute. This is the maximum amount of time that the record is kept in cache of the resolver and of the forwarding DNS. If the node goes down, the resolver still uses this value until the record expires in his cache, but as the node does not answer, the resolver tries the next name server in the list (he has the A record because he needed to know the IP of the first name server, and if he needed this IP, he already had the name server list).

Finally submit the first name server's A record to Route53, using the Create button at the bottom of the right panel:

The screenshot shows the 'Create Record Set' wizard. The 'Name' field contains 'node1.demo.francois' and the 'Type' field is set to 'A – IPv4 address'. The 'Value' field contains '34.199.174.185'. Below the value field is a note: 'IPv4 address. Enter multiple addresses on separate lines.' and an example: '192.0.2.235 198.51.100.234'. The 'Routing Policy' is set to 'Simple'. At the bottom, there's a note: 'Route 53 responds to queries based only on the values in this record. [Learn More](#)'.

Amazon, the Amazon logo, AWS and Amazon.com are trademarks of Amazon.com, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

You want (and need) to have all your cluster nodes acting as name servers, so you have to repeat these steps for all your nodes and you should get a list of A records in Route53 interface:

The screenshot shows the 'Hosted zones' section of the Route 53 console. It lists three A records under the domain 'francois'. The records are named 'node1', 'node2', and 'node3' with IP values 34.199.174.185, 34.199.175.137, and 34.199.100.46 respectively.

Name	Type	Value	Evaluate Target Health	Health Check ID	TTL	Region	Weight	Geolocation	Set ID
node1.demo.francois.demo-rlec.redislabs.com.	A	34.199.174.185	-	-	60				
node2.demo.francois.demo-rlec.redislabs.com.	A	34.199.175.137	-	-	60				
node3.demo.francois.demo-rlec.redislabs.com.	A	34.199.100.46	-	-	60				

Now, the client-side resolver and the forwarding DNS can reach the cluster nameservers by their IP address, if they know what are the names of the cluster's name servers. That's the next point.

Defining the nameserver list for a subzone

Here, the idea is to provide the list of the cluster nameserver's names to the resolvers and the forwarding DNS, so that they can resolve the IP address of one of them and query it for node's IP. To achieve that, we have to define a new record in Route53, an NS record for Name Server record. So, once again, we click on the button to *Create [a] Record Set* and we enter the relevant information in the right panel.

The name is the name of the cluster, if your cluster nodes are nodeX.mycluster.enterprise.com, then the cluster name is mycluster.enterprise.com. Remember, the resolver asks "who are the name servers for zone ". The name is the searched key, the record type is the field. In our case, the resolver asks for the 'NS' record type to get the nameservers list of the clustername, so we have to choose this type. A short TTL, such as one minute, is a good idea here, too. And we have to enter the node name list in the text box. In other DNS, we would have to create one NS record for each item, but *Route53* takes care

of that for us. I also have the habit to end these records with a final dot, it is not a typo, because some other DNS require it and it does not seem to be an issue with *Route53*. At the end, we can click on the *Create* button:

The screenshot shows the 'Create Record Set' dialog for an NS record. The 'Name' field contains 'demo.francois.' and the 'Type' field is set to 'NS – Name server'. The 'Value' field lists three name servers: 'rlec.redislabs.com.', 'node3.demo.francois.', and 'node3.demo.francois.'. The 'TTL (Seconds)' is set to 60. The 'Routing Policy' is set to 'Simple'. A note at the bottom states: 'Route 53 responds to queries based only on the values in this record. [Learn More](#)'.

Congratulations, you completed the Route53 DNS configuration for your Redis Enterprise Software. Let's check what you have.

Verification

You should end with several name server A record (one for each cluster node) to be able to reach any of them by its name. You should also have one record that lists the nameservers names for the zone (cluster):

The screenshot shows the 'Hosted zones' page with a search bar for 'francois'. It displays four record sets: one NS record for 'demo.francois.' pointing to three servers ('node1', 'node2', 'node3') and three A records for the individual nodes ('node1', 'node2', 'node3').

Name	Type	Value	Evaluate Target Health	Health Check ID	TTL	Region	Weight	Geolocation
demo.francois.	NS	node1.demo.francois.demo-rlec.redislabs.com. node2.demo.francois.demo-rlec.redislabs.com. node3.demo.francois.demo-rlec.redislabs.com.	-	-	60			
node1.demo.francois.	A	34.199.174.185	-	-	60			
node2.demo.francois.	A	34.199.175.137	-	-	60			
node3.demo.francois.	A	34.195.100.46	-	-	60			

If your cluster nodes are healthy, up and running, with DNS network ports unfiltered, you can test the configuration. Who are the nameservers in charge of the resolution in the cluster:

```

dig ns demo.francois.demo-rlec.redislabs.com

; <>> DiG 9.9.5-9+deb8u9-Debian <>> ns demo.francois.demo-rlec.redislabs.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25061
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;demo.francois.demo-rlec.redislabs.com. IN NS

;; ANSWER SECTION:
demo.francois.demo-rlec.redislabs.com. 3409 IN NS ns2.demo.francois.demo-rlec.redislabs.com.
demo.francois.demo-rlec.redislabs.com. 3409 IN NS ns1.demo.francois.demo-rlec.redislabs.com.
demo.francois.demo-rlec.redislabs.com. 3409 IN NS ns3.demo.francois.demo-rlec.redislabs.com.

;; Query time: 31 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Tue Feb 14 16:49:13 CET 2017
;; MSG SIZE rcvd: 120

```

You can see that the name are given a prefix of ns-. This answer does not come from Route53 but from the cluster nameservers themselves.

Now you can either install and configure your nodes, if not already made, or connect your client, using the cluster name (**not the IP address**).

Updated: August 11, 2022

Set up cluster behind a load balancer

When you want to setup a Redis Enterprise cluster in an environment that doesn't allow DNS, you can use a load balancer (LB) to direct traffic to the cluster nodes.

DNS role for databases

Normally, Redis Enterprise uses DNS to provide dynamic database endpoints. A DNS name such as `redis-12345.clusternname.domain` gives clients access to the database resource:

- If multiple proxies are in use, the DNS name resolves to multiple IP addresses so that clients can load balance.
- On failover or topology changes, the DNS name is automatically updated to reflect the live IP addresses.

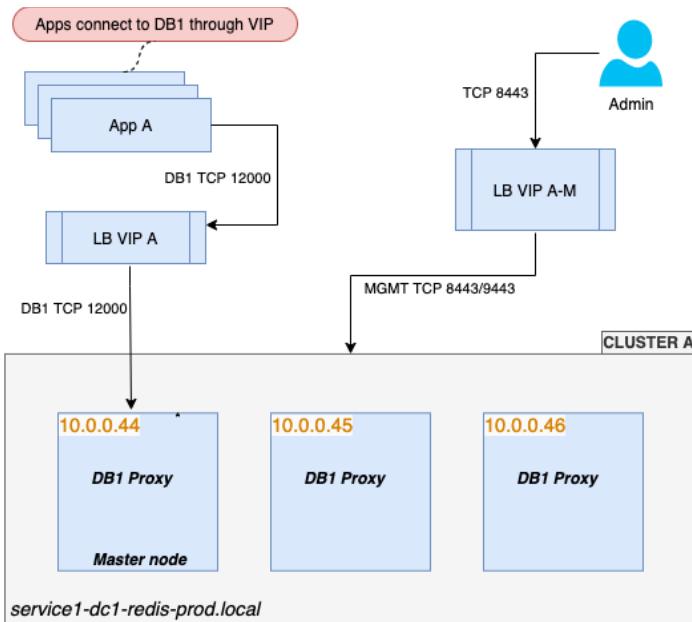
When DNS cannot be used, clients can still connect to the endpoints with the IP addresses, but the benefits of load balancing and automatic updates to IP addresses won't be available.

Network architecture with load balancer

You can compensate for the lack of DNS resolution with load balancers that can expose services and provide service discovery. A load balancer is configured in front of Redis Enterprise cluster, exposing several logical services:

- Control plane services, such as the RS admin console to access cluster administration interface
- Data plane services, such as a database endpoint to connect from client applications

Depending on which Redis Enterprise services you want to access outside the cluster you may need to configure the load balancers separately. One or more Virtual IPs (VIPs) are defined on the load balancer to expose Redis Enterprise services. The architecture is shown in the following diagram with 3 nodes Redis Enterprise cluster with one database (DB1) configured on port 12000:



Setting up an RS cluster with load balancers

Prerequisites

- [Install](#) the latest version of RS on your clusters
- Configure the cluster with the cluster name (FQDN) even though DNS is not in use. Remember that the same cluster name is used to issue the licence keys. We recommend that you use a “.local” suffix in the FQDN.

Configuring the load balancers

- Make sure that the load balancer is performing TCP health checks on the cluster nodes.
- Expose the services that you require through a Virtual IP, for example:
 - Web Management Portal (8443)
 - Rest API service (Secured - 9443; Non-secured - 8080)
 - Database ports (In the range of 10000-19999)

Other ports are shown in the list of [RS network ports](#).



Note:

Sticky, secured connections are needed only for RS admin console service (provided on port 8443).

- Certain LBAs provide specific logic to close idle connections. Either disable this feature or make sure the applications connecting to Redis use reconnection logic.
- Make sure the LB is fast enough to resolve connections between two clusters or applications that are connected to Redis databases through LB.
- Choose the standard LB which is commonly used in your environment so that you have easy access to in-house expertise for troubleshooting issues.

RS cluster configuration

There are certain recommended settings within the cluster that guarantee a flawless connectivity experience for applications and admin users when they access the cluster through a Load Balancer.



Note:

- Run the `radmin` commands directly on the cluster.
- The `radmin` commands update the settings on all nodes in the cluster.

The following settings are needed to allow inbound connections to be terminated on the relevant node inside the cluster:

```
# enable all-node proxy policy by default
rladmin tune cluster default_sharded_proxy_policy all-nodes

# ensure we redirect where necessary when running behind an LBA
rladmin cluster config handle_redirects enabled
```

An additional setting can be done to allow (on average) closer termination of client connection to where the Redis shard is located. This is an optional setting.

```
# enable sparse placement by default
rladmin tune cluster default_shards_placement sparse
```

RS database configuration

After the cluster settings are updated and the LBs are configured you can go to the RS admin console at <https://load-balancer-virtual-ip:8443> and [create a new database](#).

If you are creating an Active-Active database, you will need to use the `crdb-cli` utility. See the [crdb-cli reference](#) for more information about creating Active-Active databases from the command line.

Keep LB configuration updated when the cluster configuration changes

When your RS cluster is located behind a load balancer, you must update the LB when the cluster topology and IP addresses change. Some common cases that require you to update the LB are:

- Adding new nodes to the Redis Enterprise cluster
- Removing nodes from the Redis Enterprise cluster
- Maintenance for Redis Enterprise cluster nodes
- IP address changes for Redis Enterprise cluster nodes

After these changes, make sure that the redis connections in your applications can connect to the Redis database, especially if they are directly connected on IP addresses that have changed.

Intercluster communication considerations

Redis Enterprise supports several topologies that allow inter cluster replication, these include [Active/Passive](#) and [Active/Active](#) for deployment options. When your Redis Enterprise software clusters are located behind load balancers, you must allow some network services to be open and defined in the load balancers to allow the replication to work.

Active Passive

For Active Passive communication to work, you will need to expose database port(s) locally in each cluster (as defined above) but also allow these ports through firewalls that may be positioned between the clusters.

Active-Active

For Active-Active communication to work, you need to expose several ports, including every database port and several control plane ports as defined in [Network port configurations](#). Pay attention to services that are marked with Connection Source as "Active-Active". These ports should be allowed through firewalls that may be positioned between the clusters.

Updated: August 31, 2022

Configure cluster DNS

By default, Redis Enterprise Software deployments use DNS to communicate between nodes. You can also use the [Discovery Service](#), which uses IP addresses to connect and complies with the [Redis Sentinel API](#) supported by open source Redis.

Each node in a Redis Enterprise cluster includes a small DNS server to manage internal functions, such as high availability, automatic failover, automatic migration, and so on. Nodes should only run the DNS server included with the software. Running additional DNS servers can lead to unexpected behavior.

Cluster name and connection management

Whether you're administering Redis Enterprise Software or accessing databases, there are two ways to connect:

- URL-based connections - URL-based connections use DNS to resolve the fully qualified cluster domain name (FQDN). This means that DNS records might need to be updated when topology changes, such as adding (or removing) nodes from the cluster.

Because apps and other client connections rely on the URL (rather than the address), they do not need to be modified when topology changes.

- IP-based connections - IP-based connections do not require DNS setup, as they rely on the underlying TCP/IP addresses. As long as topology changes do not change the address of the cluster nodes, no configuration changes are needed, DNS or otherwise.

However, changes to IP addresses (or changes to IP address access) impact all connections to the node, including apps and clients. IP address changes can therefore be unpredictable or time-consuming.

URL-based connections

The fully qualified domain name (FQDN) is the unique cluster identifier that enables clients to connect to [the different components](#) of Redis Enterprise Software. The FQDN is a crucial component of the high-availability mechanism because it's used internally to enable and implement automatic and transparent failover of nodes, databases, shards, and endpoints.

 **Note:** Setting the cluster's FQDN is a one-time operation, one that cannot be changed after being set.

The FQDN must always comply with the IETF's [RFC 952](#) standard and section 2.1 of the [RFC 1123](#) standard.

Identify the cluster

To identify the cluster, either use DNS to define a fully qualified domain name or use the IP addresses of each node.

Define domain using DNS

Use DNS if you:

- have your own domain
- want to integrate the cluster into that domain
- can access and update the DNS records for that domain

1. Make sure that the cluster and at least one node (preferably all nodes) in the cluster are correctly configured in the DNS with the appropriate NS entries.

For example:

- Your domain is: `mydomain.com`
- You would like to name the Redis Enterprise Software cluster `mycluster`
- You have three nodes in the cluster:
 - `node1` (IP address `1.1.1.1`)
 - `node2` (`2.2.2.2`)
 - `node3` (`3.3.3.3`)

2. In the FQDN field, enter the value `mycluster.mydomain.com` and add the following records in the DNS table for `mydomain.com`:

<code>mycluster.mydomain.com</code>	NS	<code>node1.mycluster.mydomain.com</code>
		<code>node2.mycluster.mydomain.com</code>
		<code>node3.mycluster.mydomain.com</code>
<code>node1.mycluster.mydomain.com</code>	A	<code>1.1.1.1</code>
<code>node2.mycluster.mydomain.com</code>	A	<code>2.2.2.2</code>
<code>node3.mycluster.mydomain.com</code>	A	<code>3.3.3.3</code>

Zero-configuration using mDNS

Development and test environments can use [Multicast DNS](#) (mDNS), a zero-configuration service designed for small networks. Production environments should *not* use mDNS.

mDNS is a standard protocol that provides DNS-like name resolution and service discovery capabilities to machines on local networks with minimal to no configuration.

Before adopting mDNS, verify that it's supported by each client you wish to use to connect to your Redis databases. Also make sure that your network

infrastructure permits mDNS/multi-casting between clients and cluster nodes.

Configuring the cluster to support mDNS requires you to assign the cluster a .local name.

For example, if you want to name the Redis Enterprise Software cluster `rediscluster`, specify the FQDN name as `rediscluster.local`.

When using the DNS or mDNS option, failover can be done transparently and the DNS is updated automatically to point to the IP address of the new primary node.

IP-based connections

When you use the IP-based connection option, the FQDN does not need to have any special format because clients use IP addresses instead of hostnames to access the databases so you are free to choose whatever name you want. Using the IP-based connection option does not require any DNS configuration either.

To administer the cluster you do need to know the IP address of at least one of the nodes in the cluster. Once you have the IP address, you can simply connect to port number 8443 (for example: <https://10.0.0.12:8443>). However, as the topology of the cluster changes and node with the given IP address is removed, you need to remember the IP address of another node participating in this cluster to connect to the admin console and manage the cluster.

Applications connecting to Redis Software databases have the same constraints. When using the IP-based connection method, you can use the [Discovery Service](#) to discover the database endpoint for a given database name as long as you have an IP address for at least one of the nodes in the cluster. The API used for discovery service is compliant with the Redis Sentinel API.

To test your connection, try pinging the service. For help, see [Connect to your database](#).

Updated: August 17, 2023

Client prerequisites for mDNS



Note: mDNS is only supported for development and testing environments.

If you choose to use the mDNS protocol when [you set the cluster name](#), make sure that the configurations and prerequisites for resolving database endpoints are met on the client machines. If you have [Replica Of](#) databases on the cluster, the configurations and prerequisites are also required for the Redis Enterprise Software nodes.

To prepare a client or node for mDNS:

1. Make sure that the clients and cluster nodes are on the same physical network or have the network infrastructure configured to allow multicasting between them.
2. Install these prerequisite packages:

- For Ubuntu:

```
apt-get install libnss-mdns
```

- For RHEL/CentOS 6.x:

```
$ rpm -ivh http://download.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
$ yum install nss-mdns
$ service avahi-daemon start
```

- For RHEL/CentOS 7:

```
$ rpm -ivh https://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/e/epel-release-7-12.noarch.rpm
$ yum install nss-mdns
$ service avahi-daemon start
```

3. If you are using [mDNS with IPv6 addresses](#), update the hosts line in `/etc/nsswitch.conf` to:

```
hosts: files mdns4_minimal
\[NOTFOUND=return\] mdns
```

Multi-IP and IPv6

Redis Enterprise Software (RS) supports server/instances/VMs with multiple IP addresses, as well as IPv6 addresses.

RS related traffic can be logically and physically divided into internal traffic and external traffic:

- “Internal traffic” refers to internal cluster communications, such as communications between the nodes for cluster management purposes.
- “External traffic” refers to communications between the clients and the databases, as well as connections to the management UI in the browser.

When only one IP address exists on a machine that serves as an RS node, it is used for both internal and external traffic.

When more than one IP address exists on an RS node:

- One of the IPv4 addresses is used for internal traffic
- Other IP addresses may only be used for external traffic

As part of the node configuration process, if the machine has multiple IP addresses, you are required to assign one of the machine’s IPv4 addresses for internal traffic use, and assign one or more IPv4/IPv6 addresses for external traffic.

If at a later stage you would like to update the IP address allocation, run the relevant commands in [rladmin command-line interface \(CLI\)](#).

If you need to update the internal IP address in the OS, you must remove that node from the RS cluster, make the IP change, and then add the node back into the cluster.

When manually configuring an internal address for a node, make sure the address is valid and bound to an active interface on the node. Failure to do so prevents the node from coming back online and rejoining the cluster.

When configuring external addresses, it is possible to list external addresses that are not bound to an active interface, but are otherwise mapped or configured to route traffic to the node (AWS Elastic IPs, a load balancer VIP and so on).

`rladmin node address` commands syntax:

```
rladmin node addr set  
rladmin node external_addr set  
rladmin node external_addr [ add | remove ]
```

Where:

- `addr` - the internal address (can be used only when the node is offline)
- `external_addr` - external addresses



Note: While [joining a new node to a cluster](#) during the node bootstrap process, when prompted to provide an IP of an existing node in the cluster, if you use the node’s IP, provide the node’s internal IP address.

Network port configurations

All Redis Enterprise Software deployments span multiple physical/virtual nodes. You’ll need to keep several ports open between these nodes. This document describes the various port ranges and their uses.



Note: Whenever you create a new database, you must verify that the ports assigned to the new database’s endpoints are open. The cluster will not perform this verification for you.

Ports and port ranges used by Redis Enterprise Software

Redis Enterprise Software’s port usage falls into three general categories:

- Internal: For traffic between or within cluster nodes
- External: For traffic from client applications or external monitoring resources
- Active-Active: For traffic to and from clusters hosting Active-Active databases

Protocol	Port	Connection Source	Description
TCP	8001	Internal, External	Traffic from application to Redis Enterprise Software Discovery Service
TCP	8070, 8071	Internal, External	Metrics exported and managed by the web proxy
TCP	8443	Internal, External	Secure (HTTPS) access to the management web UI
TCP	9081	Internal	Active-Active management (internal)
TCP	9443 (Recommended), 8080	Internal, External, Active-Active	REST API traffic, including cluster management and node bootstrap
TCP	10000-19999	Internal, External, Active-Active	Database traffic
UDP	53, 5353	Internal, External	DNS/mDNS traffic
ICMP	*	Internal	Connectivity checking between nodes
TCP	1968	Internal	Proxy traffic
TCP	3333-3341, 3342-3344, 36379, 36380	Internal	Internode communication
TCP	20000-29999	Internal	Database shard traffic
TCP	8002, 8004, 8006	Internal	Default system health monitoring (envoy admin, envoy management server, gossip envoy admin)
TCP	8444, 9080	Internal	Traffic between web proxy and cnm_http/cm

Change the admin console port

The Redis Enterprise Software admin console uses port 8443, by default. You can change this to a custom port as long as the new port is not in use by another process.

To change this port, run:

```
rladmin cluster config cm_port <new-port>
```

After changing the Redis Enterprise Software web UI port, you must connect any new node added to the cluster to the UI with the custom port number: <https://newnode.mycluster.example.com:<nonstandard-port-number>>

Change the envoy ports

For system health monitoring, Redis uses the following ports by default:

- Port 8002 for envoy admin
- Port 8004 for envoy management server
- Port 8006 for gossip envoy admin

You can change each envoy port to a custom port using the `rladmin cluster config` command as long as the new port is not in use by another process. When you change `envoy_admin_port`, expect a restart of envoy.

To change the envoy admin port, run:

```
$ rladmin cluster config envoy_admin_port <new-port>
Updating envoy_admin_port... restarting now
```

To change the envoy management server port, run:

```
$ rladmin cluster config envoy_mgmt_server_port <new-port>
Cluster configured successfully
```

To change the gossip envoy admin port, run:

```
$ rladm cluster config gossip_envoy_admin_port <new-port>
Cluster configured successfully
```

Change the REST API port

For the REST API, Redis Enterprise Software uses port 9443 (secure) and port 8080 (not secure), by default. You can change this to a custom port as long as the new port is not in use by another process.

To change these ports, run:

```
rladmin cluster config cnm_http_port <new-port>
rladmin cluster config cnm_https_port <new-port>
```

Require HTTPS for API endpoints

By default, the Redis Enterprise Software API supports communication over HTTP and HTTPS. However, you can turn off HTTP support to ensure that API requests are encrypted.

Before you turn off HTTP support, make sure you migrate any scripts or proxy configurations that use HTTP to the encrypted API endpoint to prevent broken connections.

To turn off HTTP support for API endpoints, run:

```
rladmin cluster config http_support disabled
```

After you turn off HTTP support, traffic sent to the unencrypted API endpoint is blocked.

HTTP to HTTPS redirection

Starting with version 6.0.12, you cannot use automatic HTTP to HTTPS redirection. To poll metrics from the `metrics_exporter` or to access the admin console, use HTTPS in your request. HTTP requests won't be automatically redirected to HTTPS for those services.

Nodes on different VLANs

Nodes in the same cluster must reside on the same VLAN. If you can't host the nodes on the same VLAN, then you must open [all ports](#) between them.

Updated: May 11, 2023

Enable private and public database endpoints

By default, Redis Enterprise Software databases expose a single endpoint, e.g. `cluster.com` (FQDN).

When you create a cluster via the UI, you can configure it to expose private and public endpoints. This is common for environments such as cloud platforms and enterprises.

When doing so, the cluster creates an additional FQDN, e.g. `internal.cluster.com` for private network (e.g. VPC or an internal network), while the `cluster.com` FQDN can be used by a public network (e.g. the internet).

You can enable public and private endpoints during cluster creation only. However, you can still add an additional FQDN in a different domain (`cluster.io`, for example) after cluster creation.

To enable private and public endpoints:

1. Verify the IP addresses are bound to the server or instance.
2. During cluster setup, turn on **Enable public endpoints support** in the **Cluster** screen's Configuration section.

Configuration

Enable public endpoints support

When enabling public endpoints an additional internal FQDN will be created

FQDN (Fully Qualified Domain Name)

cluster.com

Private FQDN

Internal.cluster.com

If this setting is not enabled when the cluster is created, databases on the cluster support only a single endpoint.

3. Select **Next** to proceed to **Node** configuration.

4. In the **Network configuration** section:

1. Configure the machine's public IP address for external traffic.

2. Configure the private IP address for both internal and external traffic so it can be used for private database endpoints.

After cluster creation, both sets of endpoints are available for databases in the cluster.

To view and copy public and private endpoints for a database in the cluster, see the database's **Configuration > General** section.

General

Database name
Database-28-Jul-23-19-39PM

Public Endpoint
redis-<port>.cluster.com:<port> (<IP address>:<port>) [Copy](#)

Private Endpoint
redis-<port>.internal.cluster.com:<port> (<IP address>:<port>) [Copy](#)

Updated: August 17, 2023

Security

Security is an important part of any production system. This section describes the security features and settings available in Redis Enterprise.

Architecture security

When deploying Redis Enterprise Software to production, we recommend the following practices:

- **Deploy Redis Enterprise inside a trusted network:** Redis Enterprise is database software and should be deployed on a trusted network not accessible to the public internet. Deploying Redis Enterprise in a trusted network reduces the likelihood that someone can obtain unauthorized access to your data or the ability to manage your database configuration.
- **Implement anti-virus exclusions:** To ensure that anti-virus solutions that scan files or intercept processes to protect memory do not interfere with

Redis Enterprise software, customers should ensure that anti-virus exclusions are implemented across all nodes in their Redis Enterprise cluster in a consistent policy. This helps ensure that anti-virus software does not impact the availability of your Redis Enterprise cluster.

If you are replacing your existing antivirus solution or installing/supporting Redis Enterprise, make sure that the below paths are excluded:



Note: For antivirus solutions that intercept processes, binary files may have to be excluded directly depending on the requirements of your anti-virus vendor.

Path	Description
/opt/redislabs	Main installation directory for all Redis Enterprise Software binaries
/opt/redislabs/bin	Binaries for all the utilities for command line access and managements such as "rladmin" or "redis-cli"
/opt/redislabs/config	System configuration files
/opt/redislabs/lib	System library files
/opt/redislabs/sbin	System binaries for tweaking provisioning

- **Send logs to a remote logging server:** Redis Enterprise is configured to send logs by default to syslog. To send these logs to a remote logging server you must [configure syslog](#) based the requirements of the remote logging server vendor. Remote logging helps ensure that the logs are not deleted so that you can rotate the logs so that your server disk does not fill up.
- **Deploy clusters with an odd number of 3 or more nodes** - Redis is an available and partition tolerant database. We recommend that Redis Enterprise be deployed in a cluster of an odd number of 3 or more nodes so that you are able to successfully failover in the event of a failure.
- **Reboot nodes in a sequence rather than all at once:** Customers will frequently maintain reboot schedules. There are cases, however, where our customers have rebooted too many servers at once, causing a quorum failure and resulting in loss of availability of the database. We recommend that rebooting be done in a phased manner so that quorum is not lost. For example, to maintain quorum in a 3 node cluster, at least 2 nodes must be up at all times. Only one server should be rebooted at any given time to maintain quorum.
- **Implement client-side encryption:** Client-side encryption, or the practice of encrypting data within an application before storing it in a database, such as Redis, is the most widely adopted method to achieve encryption in memory. Redis is an in-memory database and stores data in-memory. If you require encryption in memory, better known as encryption in use, then client side encryption may be the right solution for you. Please be aware that when implementing solutions using client-side encryption database functions that need to operate on data — such as simple searching functions, comparisons, and incremental operations — don't work with client-side encryption.

Database security

Redis Enterprise offers several database security controls to help protect your data against unauthorized access and to improve the operational security of your database. The following section details configurable security controls available for implementation.

- **Implement role-based access for users:** With [role-based access control \(RBAC\)](#), you can [manage access control lists \(ACLs\)](#) for the entire cluster. You can reuse ACL templates across users, accounts, and multiple databases to scale complex security configurations. RBAC lets you set permissions for the Redis Enterprise Software [admin console](#), [REST API](#), and [databases](#), providing a complete security management solution for your cluster.
- **Prevent database users from logging into the admin console:** Redis Enterprise allows users to be provisioned with both control plane access and access to the database. In some scenarios this may be helpful for administrative users, but for applications we recommend that you [disable their access to the control plane](#).
- **Use strong Redis passwords:** A frequent recommendation in the security industry is to use strong passwords to authenticate users. This helps to prevent brute force password guessing attacks against your database. It's important to check that your password aligns with your organization's security policy.
- **Deactivate default user access:** Redis Enterprise comes with a "default" user for backwards compatibility with applications designed with versions of Redis prior to Redis Enterprise 6. The default user is turned on by default. This allows you to access the database without specifying a username and only using a shared secret. For applications designed to use access control lists, we recommend that you [deactivate default user access](#).
- **Enable client certificate authentication:** To prevent unauthorized access to your data, Redis Enterprise databases support the [TLS protocol](#), which includes authentication and encryption. Client certificate authentication can be used to ensure only authorized hosts can access the database.
- **Install trusted certificates:** Redis implements self-signed certificates for the database proxy and replication service, but many organizations prefer to [use their own certificates](#).
- **Configure Transport Layer Security (TLS):** Similar to the control plane, you can also [configure TLS protocols](#) to help support your security and compliance needs.
- **Configure and verify database backups:** Implementing a disaster recovery strategy is an important part of data security. Redis Enterprise supports [database backups to many destinations](#).

LDAP authentication

If your organization uses the Lightweight Directory Access Protocol (LDAP), we recommend enabling Redis Software support for role-based [LDAP authentication](#).

Updated: May 1, 2023

Access control

Grant admin console and REST API access for cluster management

Grant admin console access to a user.

Control database access using RBAC

Use role-based access control (RBAC) to configure the user's level of access to a database.

Manage users

Manage users and user security.

Manage passwords

Manage user passwords.

Role-based access control (RBAC)

An overview of role-based access control (RBAC) in Redis Enterprise Software.

LDAP authentication

Describes how Redis Enterprise Software integrates LDAP authentication and authorization. Also describes how to enable LDAP for your deployment of Redis Enterprise Software.

Updated: May 1, 2023

Grant admin console and REST API access for cluster management

To grant a user access to a Redis Enterprise cluster's admin console and [REST API](#), assign a predefined management role to the user.

Grant cluster management access

1. [Create a new user](#) or edit an existing one.
2. [Assign the user a role](#) associated with the appropriate [cluster management role](#).

Default management roles

Redis Enterprise Software includes five predefined roles that determine a user's level of access to the admin console and [REST API](#).

1. **None** - Cannot access the admin console or use the REST API
2. **DB Viewer** - Read database settings
3. **DB Member** - Administer databases
4. **Cluster Viewer** - Read cluster settings
5. **Cluster Member** - Administer the cluster
6. **Admin** - Full cluster access

For more details about the privileges granted by each of these roles, see [admin console permissions](#) or [REST API permissions](#).

Admin console permissions

Here's a summary of the admin console actions permitted by each default management role:

Action	DB Viewer	DB Member	Cluster Viewer	Cluster Member	Admin
Edit database configuration	? No	? Yes	? No	? Yes	? Yes
Reset slow log	? No	? Yes	? No	? Yes	? Yes
View cluster configuration	? No	? No	? Yes	? Yes	? Yes
View cluster logs	? No	? Yes	? Yes	? Yes	? Yes
View cluster metrics	? No	? No	? Yes	? Yes	? Yes
View database configuration	? Yes				
View database metrics	? Yes				
View node configuration	? No	? No	? Yes	? Yes	? Yes
View node metrics	? No	? No	? Yes	? Yes	? Yes
View Redis database password	? No	? Yes	? No	? Yes	? Yes
View and edit cluster settings	? No	? No	? No	? No	? Yes

More info

- [Add users](#)
- [Role-based access control \(RBAC\) overview](#)

Updated: August 17, 2023

Control database access using RBAC

Default user

When you create a database, [default user access](#) is enabled by default.

If you set up [role-based access controls](#) for your database and don't require compatibility with versions earlier than Redis 6, you can [deactivate the default user](#).



Warning - Before you [deactivate default user access](#), make sure the role associated with the database is [assigned to a user](#). Otherwise, the database will be inaccessible.

Role-based access control

To control a user's level of access to a database, use [role-based access control \(RBAC\)](#).

1. [Configure Redis ACLs](#).
2. [Create or edit a role](#) and associate it with specific Redis ACLs and databases.
3. [Assign the role to a user](#) to grant access to the database with the permissions defined by the role's associated ACLs.

More info

- [Add users](#)

- [Role-based access control \(RBAC\) overview](#)
-

Updated: May 1, 2023

Manage users

Redis Enterprise supports the following user account security settings:

- Password complexity
- Password expiration
- User lockouts
- Account inactivity timeout

Manage users and user security

Add users

Add users to the cluster and assign access control roles (ACLs) to them.

Manage user login

Manage user login lockout and session timeout.

Manage default user

Manage a database's default user.

Updated: June 21, 2022

Add users

To add a user to the cluster:

1. From the **Access Control > Users** tab in the admin console, select **+ Add user**.
2. Enter the name, email, and password of the new user.
3. Assign a **Role** to the user to grant permissions for cluster management and data access.
4. Select the **Alerts** the user should receive by email:
 - **Receive alerts for databases** - The alerts that are enabled for the selected databases will be sent to the user. Choose **All databases** or **Customize** to select the individual databases to send alerts for.
 - **Receive cluster alerts** - The alerts that are enabled for the cluster in **Cluster > Alerts Settings** are sent to the user.
5. Select **Save**.

More info

- [Grant admin console and REST API access for cluster management](#)
 - [Control database access using RBAC](#)
-

Updated: August 17, 2023

Manage user login

Redis Enterprise Software secures user access in a few different ways, including automatically:

- Locking user accounts after a series of authentication failures (invalid passwords)
- Signing sessions out after a period of inactivity

Here, you learn how to configure the relevant settings.

User login lockout

The parameters for the user login lockout are:

- **Login Lockout Threshold** - The number of failed login attempts allowed before the user account is locked. (Default: 5 minutes)
- **Login Lockout Counter Reset** - The amount of time during which failed login attempts are counted. (Default: 15 minutes)
- **Login Lockout Duration** - The amount of time that the user account is locked after excessive failed login attempts. (Default: 30 minutes)

By default, after 5 failed login attempts within 15 minutes, the user account is locked for 30 minutes.

You can view the user login restrictions for your cluster with:

```
rladmin info cluster | grep login_lockout
```

Change the login lockout threshold

You can set the login lockout threshold with the command:

```
rladmin tune cluster login_lockout_threshold <login_lockout_threshold>
```

For example, to set the lockout threshold to 10 failed login attempts, run:

```
rladmin tune cluster login_lockout_threshold 10
```

If you set the lockout threshold to 0, it turns off account lockout. In this case, the cluster settings show `login_lockout_threshold: disabled`.

Change the login lockout counter

You can set the login lockout reset counter in seconds with the command:

```
rladmin tune cluster login_lockout_counter_reset_after <login_lockout_counter_reset_after>
```

To set the lockout reset to 1 hour, run:

```
rladmin tune cluster login_lockout_counter_reset_after 3600
```

Change the login lockout duration

You can set the login lockout duration in seconds with the command:

```
rladmin tune cluster login_lockout_duration <login_lockout_duration>
```

For example, to set the lockout duration to 1 hour, run:

```
rladmin tune cluster login_lockout_duration 3600
```

If you set the lockout duration to 0, then the account can be unlocked only when an administrator changes the account's password. In this case, the cluster settings show `login_lockout_duration: admin-release`.

Unlock locked user accounts

To unlock a user account or reset a user password with `rladmin`, run:

```
rladmin cluster reset_password <user_email>
```

To unlock a user account or reset a user password with the REST API, use [PUT /v1/users](#):

```
PUT https://[host][:port]/v1/users  
'{"password": "<new_password>"}'
```

Session timeout

The Redis Enterprise admin console supports session timeouts. By default, users are automatically logged out after 15 minutes of inactivity.

To customize the session timeout, run:

```
r1admin cluster config cm_session_timeout_minutes <number_of_min>
```

The `number_of_min` is the number of minutes after which sessions will time out.

Updated: August 10, 2022

Manage default user

When you [create a database](#), default user database access is enabled by default (**Unauthenticated access** is selected). This gives the default user full access to the database and enables compatibility with versions of Redis before Redis 6.

Select **Password-only authentication**, then enter and confirm a default database password to require authentication for connections to the database.

Access Control

Access method

Unauthenticated access
You can access the database as the default user without providing credentials.

Password-only authentication
You must provide the password to access the database as the default user.

Using ACL only
To access the database, you must associate at least one role and ACL with the database.

Enter Password 

Re-Enter Password 

Access Control List

[+ Add ACL](#)

Authenticate as default user

When you configure a password for your database, all connections to the database must authenticate using the [AUTH](#) command.

AUTH <default-database-password>

Change default database password

To change the default user's password:

1. From the database's **Security** tab, select **Edit**.
2. In the **Access Control** section, select **Password-only authentication** as the **Access method**.
3. Enter and re-enter the new password.
4. Select **Save**.

Deactivate default user

If you set up [role-based access control](#) with [access control lists](#) (ACLs) for your database and don't require backwards compatibility with versions earlier than Redis 6, you can [deactivate the default user](#).

 **Warning** - Before you deactivate default user access, make sure the role associated with the database is [assigned to a user](#). Otherwise, the database will be inaccessible.

To deactivate the default user:

1. From the database's **Security** tab, select **Edit**.
2. In the **Access Control** section, select **Using ACL only** as the **Access method**.

Access Control

Access method

Unauthenticated access
You can access the database as the default user without providing credentials.

Password-only authentication
You must provide the password to access the database as the default user.

Using ACL only
To access the database, you must associate at least one role and ACL with the database.

Access Control List

Role name	Users	Redis ACLs
<input type="button" value="Admin"/>	<input type="button" value="Administrator"/>	<input type="button" value="Full Access"/>

+ Add ACL

3. Choose at least one role and Redis ACL to access the database.

4. Select **Save**.

Updated: August 17, 2023

Manage passwords

Redis Enterprise Software provides several ways to manage the passwords of local accounts, including:

- [Password complexity rules](#)
- [Password expiration](#)
- [Password rotation policies](#)

You can also manage a user's ability to [sign in](#) and control [session timeout](#).

To enforce more advanced password policies, we recommend using [LDAP integration](#) with an external identity provider, such as Active Directory.

Updated: May 1, 2023

Enable password complexity rules

Redis Enterprise Software provides optional password complexity rules that meet common requirements. When enabled, these rules require the password to have:

- At least 8 characters
- At least one uppercase character
- At least one lowercase character
- At least one number
- At least one special character

These requirements reflect v6.2.12 and later. Earlier versions did not support numbers or special characters as the first or the last character of a password. This restriction was removed in v6.2.12.

In addition, the password:

- Cannot contain the user's email address or the reverse of the email address.
- Cannot have more than three repeating characters.

Password complexity rules apply when a new user account is created and when the password is changed. Password complexity rules are not applied to accounts authenticated by an external identity provider.

You can use the admin console or the REST API to enable password complexity rules.

Enable using the admin console

To enable password complexity rules using the admin console:

1. Go to **Cluster > Security > Preferences**, then select **Edit**.
2. In the **Password** section, turn on **Complexity rules**.
3. Select **Save**.

Enable using the REST API

To use the REST API to enable password complexity rules:

```
PUT https://[host][:port]/v1/cluster
{"password_complexity":true}
```

Deactivate password complexity rules

To deactivate password complexity rules:

- Use the admin console:
 1. Go to **Cluster > Security > Preferences**, then select **Edit**.

2. In the **Password** section, turn off **Complexity rules**.
 3. Select **Save**.
- Use the `cluster` REST API endpoint to set `password_complexity` to `false`

Updated: August 17, 2023

Configure password expiration

Enable password expiration

To enforce an expiration of a user's password after a specified number of days:

- Use the admin console:
 1. Go to **Cluster > Security > Preferences**, then select **Edit**.
 2. In the **Password** section, turn on **Expiration**.
 3. Enter the number of days before passwords expire.
 4. Select **Save**.
- Use the `cluster` endpoint of the REST API

```
PUT https://[host][:port]/v1/cluster
{"password_expiration_duration":<number_of_days>}
```

Deactivate password expiration

To deactivate password expiration:

- Use the admin console:
 1. Go to **Cluster > Security > Preferences**, then select **Edit**.
 2. In the **Password** section, turn off **Expiration**.
 3. Select **Save**.
- Use the `cluster` REST API endpoint to set `password_expiration_duration` to 0 (zero).

Updated: August 17, 2023

Rotate passwords

Redis Enterprise Software lets you implement password rotation policies using the [REST API](#).

You can add a new password for a database user without immediately invalidating the old one (which might cause authentication errors in production).

 | Note: Password rotation does not work for the default user. [Add additional users](#) to enable password rotation.

Password rotation policies

For user access to the Redis Enterprise Software admin console, you can set a [password expiration policy](#) to prompt the user to change their password.

However, for database connections that rely on password authentication, you need to allow for authentication with the existing password while you roll out the new password to your systems.

With the Redis Enterprise Software REST API, you can add additional passwords to a user account for authentication to the database or the admin console and API.

After the old password is replaced in the database connections, you can delete the old password to finish the password rotation process.

 **Warning** - Multiple passwords are only supported using the REST API. If you reset the password for a user in the admin console, the new password replaces all other passwords for that user.

The new password cannot already exist as a password for the user and must meet the [password complexity](#) requirements, if enabled.

Rotate password

To rotate the password of a user account:

1. Add an additional password to a user account with [POST /v1/users/password](#):

```
POST https://[host][:port]/v1/users/password
'{"username":<username>, "old_password":<an_existing_password>, "new_password":<a_new_password>}'
```

After you send this request, you can authenticate with both the old and the new password.

2. Update the password in all database connections that connect with the user account.
3. Delete the original password with [DELETE /v1/users/password](#):

```
DELETE https://[host][:port]/v1/users/password
'{"username":<username>, "old_password":<an_existing_password>}'
```

If there is only one valid password for a user account, you cannot delete that password.

Replace all passwords

You can also replace all existing passwords for a user account with a single password that does not match any existing passwords. This can be helpful if you suspect that your passwords are compromised and you want to quickly resecure the account.

To replace all existing passwords for a user account with a single new password, use [PUT /v1/users/password](#):

```
PUT https://[host][:port]/v1/users/password
'{"username":<username>, "old_password":<an_existing_password>, "new_password":<a_new_password>}'
```

All of the existing passwords are deleted and only the new password is valid.

 **Note:** If you send the above request without specifying it is a PUT request, the new password is added to the list of existing passwords.

Updated: May 1, 2023

Update admin credentials for Active-Active databases

Active-Active databases use administrator credentials to manage operations.

To update the administrator user password on a cluster with Active-Active databases:

1. From the user management page, update the administrator user password on the clusters you want to update.
2. For each participating cluster *and* each Active-Active database, update the admin user credentials to match the changes in step 1.

 **Warning** - Do not perform any management operations on the databases until these steps are complete.

Updated: May 1, 2023

Role-based access control (RBAC)

Role-based access control (RBAC) allows you to configure the level of access each user has to a Redis Enterprise cluster's [admin console](#), [REST API](#), and [databases](#). To grant permissions, assign [predefined](#) or custom roles to a user. You can create a role once and then deploy it across multiple databases in the cluster.

Role types

You can create custom user roles that determine cluster management permissions, data access permissions, or a combination of both.

- [Management roles](#) determine user access to the cluster's admin console and [REST API](#).
- [Data access controls](#) determine the permissions each role grants for each database in the cluster.

Multiple users can share the same role.

Access control screen

The Access Control screen has the following tabs:

- **Users** - [Create users](#) and [assign a role to each user](#) to grant access to the admin console, REST API, or databases.
- **Roles** - [Create roles](#). Each role consists of a set of permissions (Redis ACLs) for one or more Redis databases. You can reuse these roles for multiple users.
- **Redis ACLs** - [Define named permissions](#) for specific Redis commands, keys, and pub/sub channels. Redis version 7.2 lets you specify read and write access for key patterns and use selectors to define multiple sets of rules in a single Redis ACL. You can use defined Redis ACLs for multiple databases and roles.
- **LDAP Mappings** - Map LDAP groups to access control roles.
- **Settings** - Additional access control settings, such as default permissions for pub/sub ACLs.

Active-Active databases

Users, roles, and Redis ACLs are cluster-level entities, which means:

- They apply to the local participating cluster and Active-Active database instance.
- They do not replicate or propagate to the other participating clusters and instances.
- ACLs are enforced according to the instance connected to the client. The Active-Active replication mechanism propagates all the effects of the operation.

More info

- [Grant admin console and REST API access for cluster management](#)
- [Control database access using RBAC](#)
- [Redis ACL rules](#)

Updated: August 17, 2023

Configure ACLs to define database permissions

Redis access control lists (Redis ACLs) allow you to define named permissions for specific Redis commands, keys, and pub/sub channels. You can use defined Redis ACLs for multiple databases and roles.

Predefined Redis ACLs

Redis Enterprise Software provides one predefined Redis ACL named **Full Access**. This ACL allows all commands on all keys and cannot be edited.

Redis ACL syntax

Redis ACLs are defined by a [Redis syntax](#) where you specify the commands or command categories that are allowed for specific keys.

Commands and categories

Redis ACL rules can allow or block specific [Redis commands](#) or [command categories](#).

- + includes commands
- - excludes commands
- +@ includes command categories
- -@ excludes command categories

The following example allows all read commands and the SET command:

```
+@read +SET
```

Module commands have several ACL limitations:

- [Redis modules](#) do not have command categories.
- Other [command category](#) ACLs, such as +@read and +@write, do not include Redis module commands. +@all is the only exception because it allows all Redis commands.
- You have to include individual module commands in a Redis ACL rule to allow them.

For example, the following Redis ACL rule allows read-only commands and the RediSearch commands FT . INFO and FT . SEARCH:

```
+@read +FT.INFO +FT.SEARCH
```

Key patterns

To define access to specific keys or key patterns, use the following prefixes:

- ~ or %RW~ allows read and write access to keys.
- %R~ allows read access to keys.
- %W~ allows write access to keys.

%RW~, %R~, and %W~ are only supported for databases with Redis version 7.2 or later.

The following example allows read and write access to all keys that start with “app1” and read-only access to all keys that start with “app2”:

```
~app1* %R~app2*
```

Pub/sub channels

The & prefix allows access to [pub/sub channels](#) (only supported for databases with Redis version 6.2 or later).

To limit access to specific channels, include `resetchannels` before the allowed channels:

```
resetchannels &channel1 &channel2
```

Selectors

[Selectors](#) let you define multiple sets of rules in a single Redis ACL (only supported for databases with Redis version 7.2 or later). A command is allowed if it matches the base rule or any selector in the Redis ACL.

- (`<rule set>`) creates a new selector.
- `clearselectors` deletes all existing selectors for a user. This action does not delete the base ACL rule.

In the following example, the base rule allows GET key1 and the selector allows SET key2:

```
+GET ~key1 (+SET ~key2)
```

Configure Redis ACLs

To configure a Redis ACL rule that you can assign to a user role:

1. From **Access Control > Redis ACLs**, you can either:

- Point to a Redis ACL and select  to edit an existing Redis ACL.
- Select **+ Add Redis ACL** to create a new Redis ACL.

2. Enter a descriptive name for the Redis ACL. This will be used to reference the ACL rule to the role.

3. Define the ACL rule.



Note: The ACL builder does not support selectors and key permissions. Use [Free text command](#) to manually define them instead.

4. Select **Save**.

Note: For multi-key commands on multi-slot keys, the return value is `failure`, but the command runs on the keys that are allowed.

Default pub/sub permissions

Redis database version 6.2 introduced pub/sub ACL rules that determine which [pub/sub channels](#) a user can access.

The configuration option `acl-pubsub-default`, added in Redis Enterprise Software version 6.4.2, determines the cluster-wide default level of access for all pub/sub channels. Redis Enterprise Software uses the following pub/sub permissions by default:

- For versions 6.4.2 and 7.2, `acl-pubsub-default` is permissive (`allchannels` or `&*`) by default to accommodate earlier Redis versions.
- In future versions, `acl-pubsub-default` will change to restrictive (`resetchannels`). Restrictive permissions block all pub/sub channels by default, unless explicitly permitted by an ACL rule.

If you use ACLs and pub/sub channels, you should review your databases and ACL settings and plan to transition your cluster to restrictive pub/sub permissions in preparation for future Redis Enterprise Software releases.

Prepare for restrictive pub/sub permissions

To secure pub/sub channels and prepare your cluster for future Redis Enterprise Software releases that default to restrictive pub/sub permissions:

1. Upgrade Redis databases:

- For Redis Enterprise Software version 6.4.2, upgrade all databases in the cluster to Redis DB version 6.2.
- For Redis Enterprise Software version 7.2, upgrade all databases in the cluster to Redis DB version 7.2 or 6.2.

2. Create or update ACLs with permissions for specific channels using the `resetchannels &channel` format.

3. Associate the ACLs with relevant databases.

4. Set default pub/sub permissions (`acl-pubsub-default`) to restrictive. See [Change default pub/sub permissions](#) for details.

5. If any issues occur, you can temporarily change the default pub/sub setting back to permissive. Resolve any problematic ACLs before making pub/sub permissions restrictive again.



Note:

When you change the cluster's default pub/sub permissions to restrictive, &* is added to the **Full Access** ACL. Before you make this change, consider the following:

- Because pub/sub ACL syntax was added in Redis 6.2, you can't associate the **Full Access** ACL with database versions 6.0 or lower after this change.
- The **Full Access** ACL is not reverted if you change `acl-pubsub-default` to permissive again.
- Every database with the default user enabled uses the **Full Access** ACL.

Change default pub/sub permissions

As of Redis Enterprise version 6.4.2, you can configure `acl_pubsub_default`, which determines the default pub/sub permissions for all databases in the cluster. You can set `acl_pubsub_default` to the following values:

- `resetchannels` is restrictive and blocks access to all channels by default.
- `allchannels` is permissive and allows access to all channels by default.

To make default pub/sub permissions restrictive:

1. [Upgrade all databases](#) in the cluster to Redis version 6.2 or later.
2. Set the default to restrictive (`resetchannels`) using one of the following methods:
 - New admin console (only available for Redis Enterprise versions 7.2 and later):
 1. Navigate to **Access Control** > **Settings** > **Pub/Sub ACLs** and select **Edit**.
 2. For **Default permissions for Pub/Sub ACLs**, select **Restrictive**, then **Save**.
 - `radmin tune cluster`:

```
radmin tune cluster acl_pubsub_default resetchannels
```
 - [Update cluster policy](#) REST API request:

```
PUT /v1/cluster/policy
{ "acl_pubsub_default": "resetchannels" }
```

ACL command support

Redis Enterprise Software does not support certain open source Redis ACL commands. Instead, you can manage access controls from the admin console.

Command	Supported
ACL CAT	Supported
ACL DELUSER	Not supported
ACL DRYRUN	Supported
ACL GENPASS	Not supported
ACL GETUSER	Supported
ACL HELP	Supported
ACL LIST	Supported
ACL LOAD	Not supported
ACL LOG	Not supported
ACL SAVE	Not supported
ACL SETUSER	Not supported
ACL USERS	Supported
ACL WHOAMI	Supported

Redis ACLs also have the following differences in Redis Enterprise Software:

- The MULTI, EXEC, DISCARD commands are always allowed, but ACLs are enforced on MULTI subcommands.
- Nested selectors are not supported.

For example, the following selectors are not valid in Redis Enterprise: +GET ~key1 (+SET (+SET ~key2) ~key3)

- Key and pub/sub patterns do not allow the following characters: ' (', ') '
- The following password configuration syntax is not supported: '>', '<', '#!', 'resetpass'

To configure passwords in Redis Enterprise Software, use one of the following methods:

- `rladmin cluster reset_password`:

```
rladmin cluster reset_password <user email>
```

- REST API `PUT /v1/users` request and provide password

Next steps

- [Create or edit a role](#) and add Redis ACLs to it.

Updated: August 17, 2023

Create roles

From [Access Control > Roles](#), you can create custom user roles that determine cluster management permissions, data access permissions, or a combination of both.

- **Management roles** - Management roles define user access to the cluster's admin console and API.
- **Data access controls** - Data access controls define the permissions each role has for each database in the cluster.

Default management roles

Redis Enterprise Software includes five predefined roles that determine a user's level of access to the admin console and [REST API](#).

1. **None** - Cannot access the admin console or use the REST API
2. **DB Viewer** - Read database settings
3. **DB Member** - Administer databases
4. **Cluster Viewer** - Read cluster settings
5. **Cluster Member** - Administer the cluster
6. **Admin** - Full cluster access

For more details about the privileges granted by each of these roles, see [admin console permissions](#) or [REST API permissions](#).

Create roles for database access

To create a role that grants database access to users but blocks access to the Redis Enterprise admin console and REST API, set the [Cluster management role](#) to **None**.

To define a role for database access:

1. From [Access Control > Roles](#), you can:
 - Point to a role and select  to edit an existing role.
 - Select **+ Add role** to create a new role.
2. Enter a descriptive name for the role. This will be used to reference the role when configuring users.
3. Choose a **Cluster management role**. The default is **None**.
4. Select **+ Add ACL**.

5. Choose a Redis ACL and databases to associate with the role.
6. Select the check mark to confirm.
7. Select **Save**.

Next steps

- [Assign the role to a user](#).

Updated: August 17, 2023

Assign roles to users

Assign a role, associated with specific databases and access control lists (ACLs), to a user to grant database access:

1. From the **Access Control > Users** tab in the admin console, you can:
 - Point to an existing user and select  to edit the user.
 - Select **+ Add user** to [create a new user](#).
2. Select a role to assign to the user.
3. Select **Save**.

Next steps

Depending on the type of the user's assigned role (cluster management role or data access role), the user can now:

- [Connect to a database](#) associated with the role and run limited Redis commands, depending on the role's Redis ACLs.
- Sign in to the Redis Enterprise Software admin console.
- Make a [REST API](#) request.

Updated: August 17, 2023

LDAP authentication

Redis Enterprise Software supports [Lightweight Directory Access Protocol](#) (LDAP) authentication and authorization through its [role-based access controls](#) (RBAC). You can use LDAP to authorize access to the admin console and to control database access.

You can configure LDAP roles using the Redis Enterprise admin console or REST API.

How it works

Here's how role-based LDAP integration works:

1. A user signs in with their LDAP credentials.

Based on the LDAP configuration details, the username is mapped to an LDAP Distinguished Name.
2. A simple [LDAP bind request](#) is attempted using the Distinguished Name and the password. The sign-in fails if the bind fails.
3. Obtain the user's LDAP group memberships.

Using configured LDAP details, obtain a list of the user's group memberships.
4. Compare the user's LDAP group memberships to those mapped to local roles.
5. Determine if one of the user's groups is authorized to access the target resource. If so, the user is granted the level of access authorized to the role.

To access the admin console, the user needs to belong to an LDAP group mapped to an administrative role.

For database access, the user needs to belong to an LDAP group mapped to a role listed in the database's access control list (ACL). The rights granted to the group determine the user's level of access.

Prerequisites

Before you enable LDAP in Redis Enterprise, you need:

1. The LDAP groups that correspond to the levels of access you wish to authorize. Each LDAP group will be mapped to a Redis Enterprise access control role.
2. A Redis Enterprise access control role for each LDAP group. Before you enable LDAP, you need to set up [role-based access controls \(RBAC\)](#).
3. The following LDAP details:
 - Server URI, including host, port, and protocol details.
 - Certificate details for secure protocols.
 - Bind credentials, including Distinguished Name, password, and (optionally) client public and private keys for certificate authentication.
 - Authentication query details, whether template or query.
 - Authorization query details, whether attribute or query.
 - The Distinguished Names of LDAP groups you'll use to authorize access to Redis Enterprise resources.

Enable LDAP

To enable LDAP:

1. From **Cluster > Security > LDAP** in the admin console, [enable LDAP access](#).
2. Map LDAP groups to [access control roles](#).
3. Update database access control lists (ACLs) to [authorize role access](#).

If you already have appropriate roles, you can update them to include LDAP groups.

More info

- Enable and configure [role-based LDAP](#)
- Map LDAP groups to [access control roles](#)
- Update database ACLs to [authorize LDAP access](#)
- Learn more about Redis Enterprise Software [security and practices](#)

Updated: August 17, 2023

Enable role-based LDAP

Redis Enterprise Software uses a role-based mechanism to enable LDAP authentication and authorization.

When a user attempts to access Redis Enterprise resources using LDAP credentials, the credentials are passed to the LDAP server in a bind request. If the request succeeds, the user's groups are searched for a group that authorizes access to the original resource.

Role-based LDAP lets you authorize admin console admins (previously known as *external users*) as well as database users. As with any access control role, you can define the level of access authorized by the role.

Set up LDAP connection

To enable and configure LDAP, sign into the Redis Enterprise admin console and then select **Cluster > Security > LDAP**.

Cluster active Configuration Security Metrics Logs Modules

Certificates **LDAP** Preferences OCSP

Cancel **Save & enable**

LDAP Server

Protocol **LDAP** LDAPS START-TLS

Host Port
ldap:// 389 ✓ ✕

+ Add host

Bind Credentials

Distinguished Name Password

Authentication Query

Search user by: **Query** Template

Base Filter Scope
baseObject

Authorization Query

Search groups by: **Query** Attribute

Base Filter Scope
baseObject

When LDAP is enabled, use the info you gathered to populate the following settings.

LDAP server settings

The **LDAP Server** settings define the communication settings used for LDAP authentication and authorization. These include:

Setting	Description
Protocol	Underlying communication protocol; must be <i>LDAP</i> , <i>LDAPS</i> , or <i>STARTTLS</i>
Host	URL of the LDAP server
Port	LDAP server port number
Trusted CA certificate	(<i>LDAPS</i> or <i>STARTTLS</i> protocols only) Certificate for the trusted certificate authority (CA)

When defining multiple LDAP hosts, the organization tree structure must be identical for all hosts.

Bind credentials

These settings define the credentials for the bind query:

Setting	Description
Distinguished Name	Example: cd=admin,dc=example,dc=org

<i>Setting</i>	<i>Description</i>
Password	Example: admin1
Client certificate authentication	(LDAPS or STARTTLS protocols only) Place checkmark to enable
Client public key	(LDAPS or STARTTLS protocols only) The client public key for authentication
Client private key	(LDAPS or STARTTLS protocols only) The client private key for authentication

Authentication query

These settings define the authentication query:

<i>Setting</i>	<i>Description</i>
Search user by	Either Template or Query
Template	(template search) Example: cn=%u, ou=dev, dc=example, dc=com
Base	(query search) Example: ou=dev, dc=example, dc=com
Filter	(query search) Example: (cn=%u)
Scope	(query search) Must be baseObject, singleLevel, or wholeSubtree

In this example, %u is replaced by the username attempting to access the Redis Enterprise resource.

Authorization query

These settings define the group authorization query:

<i>Setting</i>	<i>Description</i>
Search groups by	Either Attribute or Query
Attribute	(attribute search) Example: memberOf (case-sensitive)
Base	(query search) Example: ou=groups, dc=example, dc=com
Filter	(query search) Example: (members=%D)
Scope	(query search) Must be baseObject, singleLevel, or wholeSubtree

In this example, %D is replaced by the Distinguished Name of the user attempting to access the Redis Enterprise resource.

Save settings

When finished, select the **Save & enable** button to save your changes.

More info

- Map LDAP groups to [access control roles](#)
- Update database ACLs to [authorize LDAP access](#)
- Learn more about Redis Software [security and practices](#)

Updated: August 17, 2023

Map LDAP groups to roles

Redis Enterprise Software uses a role-based mechanism to enable LDAP authentication and authorization.

Once LDAP is enabled, you need to map LDAP groups to Redis Enterprise access control roles.

Map LDAP groups to roles

To map LDAP groups to access control roles:

1. From the admin console menu, select **Access Control > LDAP Mappings**.

If you see an “LDAP configuration is off or disabled” message, go to **Cluster > Security > LDAP** to [enable role-based LDAP](#).

You can map LDAP roles when LDAP configuration is not enabled, but they won’t have any effect until you configure and enable LDAP.

2. Select the **+ Add LDAP Mapping** button to create a new mapping and then enter the following details:

Setting	Description
Name	A descriptive, unique name for the mapping
Distinguished Name	The distinguished name of the LDAP group to be mapped. Example: cn=admins, ou=groups, dc=example, dc=com
Role	The Redis Software access control role defined for this group
Email	(Optional) An address to receive alerts
Alerts	Selections identifying the desired alerts.

3. When finished, select the **Save** button.

Create a mapping for each LDAP group used to authenticate and/or authorize access to Redis Enterprise Software resources.

The scope of the authorization depends on the access control role:

- If the role authorizes admin management, LDAP users are authorized as admin console administrators.
- If the role authorizes database access, LDAP users are authorized to use the database to the limits specified in the role.
- To authorize LDAP users to specific databases, update the database access control lists (ACLs) to include the mapped LDAP role.

More info

- Enable and configure [role-based LDAP](#)
- Update database ACLs to [authorize LDAP access](#)
- Learn more about Redis Enterprise Software [security and practices](#)

Updated: August 17, 2023

Update database ACLs

To grant LDAP users access to a database, assign the mapped access role to the access control list (ACL) for the database.

1. From the admin console menu, select **Databases** and then select the database from the list.
2. From the **Security** tab, select the **Edit** button.
3. In the **Access Control List** section, select **+ Add ACL**.

Access Control

^

Access method

Unauthenticated access

You can access the database as the default user without providing credentials.

Password-only authentication

You must provide the password to access the database as the default user.

Using ACL only

To access the database, you must associate at least one role and ACL with the database.

Access Control List

Role name	Users	Redis ACLs	
 Admin ▼	Administrator	Full Access ▼	✓ ✖
+ Add ACL			

4. Select the appropriate roles and then save your changes.

If you assign multiple roles to an ACL and a user is authorized by more than one of these roles, their access is determined by the first “matching” rule in the list.

If the first rule gives them read access and the third rule authorizes write access, the user will only be able to read data.

As a result, we recommend ordering roles so that higher access roles appear before roles with more limited access.

More info

- Enable and configure [role-based LDAP](#)
- Map LDAP groups to [access control roles](#)
- Learn more about Redis Enterprise Software [security and practices](#)

Updated: August 17, 2023

Migrate to role-based LDAP

Redis Enterprise Software supports LDAP through a [role-based mechanism](#), first introduced in [v6.0.20](#).

Earlier versions of Redis Enterprise Software supported a cluster-based mechanism; however, that mechanism was removed in v6.2.12.

If you’re using the cluster-based mechanism to enable LDAP authentication, you need to migrate to the role-based mechanism before upgrading to Redis Enterprise Software v6.2.12 or later.

Migration checklist

This checklist covers the basic process:

1. Identify accounts per app on the customer end.
2. Create or identify an LDAP user account on the server that is responsible for LDAP authentication and authorization.
3. Create or identify an LDAP group that contains the app team members.

4. Verify or configure the Redis Enterprise ACLs.
5. Configure each database ACL.
6. Remove the earlier “external” (LDAP) users from Redis Enterprise.
7. *(Recommended)* Update cluster configuration to replace the cluster-based configuration file.

You can use `radmin` to update the cluster configuration:

```
$ touch /tmp/saslauthd_empty.conf
$ radmin cluster config saslauthd_ldap_conf \
  /tmp/saslauthd_empty.conf
```

Here, a blank file replaces the earlier configuration.

8. Use **Cluster > Security > LDAP** to enable role-based LDAP.
9. Map your LDAP groups to access control roles.
10. Test application connectivity using the LDAP credentials of an app team member.
11. *(Recommended)* Turn off default access for the database to avoid anonymous client connections.

Because deployments and requirements vary, you'll likely need to adjust these guidelines.

Test LDAP access

To test your LDAP integration, you can:

- Connect with `redis-cli` and use the [AUTH command](#) to test LDAP username/password credentials.
- Sign in to the admin console using LDAP credentials authorized for admin access.
- Use [RedisInsight](#) to access a database using authorized LDAP credentials.
- Use the [REST API](#) to connect using authorized LDAP credentials.

More info

- Enable and configure [role-based LDAP](#)
- Map LDAP groups to [access control roles](#)
- Update database ACLs to [authorize LDAP access](#)
- Learn more about Redis Enterprise Software [security and practices](#)

Updated: August 17, 2023

Audit connection events

Starting with version 6.2.18, Redis Enterprise Software lets you audit database connection and authentication events. This helps you track and troubleshoot connection activity.

The following events are tracked:

- Database connection attempts
- Authentication requests, including requests for new and existing connections
- Database disconnects

When tracked events are triggered, notifications are sent via TCP to an address and port defined when auditing is enabled. Notifications appear in near real time and are intended to be consumed by an external listener, such as a TCP listener, third-party service, or related utility.

For development and testing environments, notifications can be saved to a local file; however, this is neither supported nor intended for production environments.

For performance reasons, auditing is not enabled by default. In addition, auditing occurs in the background (asynchronously) and is non-blocking by design. That is, the action that triggered the notification continues without regard to the status of the notification or the listening tool.

Enable audit notifications

Cluster audits

To enable auditing for your cluster, use:

- `rladmin`

```
rladmin cluster config auditing db_conns \
  audit_protocol <TCP|local> \
  audit_address <address> \
  audit_port <port> \
  audit_reconnect_interval <interval in seconds> \
  audit_reconnect_max_attempts <number of attempts>
```

where:

- *audit_protocol* indicates the protocol used to process notifications. For production systems, *TCP* is the only value.
- *audit_address* defines the TCP/IP address where one can listen for notifications
- *audit_port* defines the port where one can listen for notifications
- *audit_reconnect_interval* defines the interval (in seconds) between attempts to reconnect to the listener. Default is 1 second.
- *audit_reconnect_max_attempts* defines the maximum number of attempts to reconnect. Default is 0. (infinite)

Development systems can set *audit_protocol* to *local* for testing and training purposes; however, this setting is not supported for production use.

When *audit_protocol* is set to *local*, *<address>* should be set to a [stream socket](#) defined on the machine running Redis Enterprise and *<port>* should not be specified:

```
rladmin cluster config auditing db_conns \
  audit_protocol local audit_address <output-file>
```

The output file (and path) must be accessible by the user and group running Redis Enterprise Software.

- the [REST API](#)

```
PUT /v1/cluster/auditing/db_conns
{
  "audit_address": "<address>",
  "audit_port": <port>,
  "audit_protocol": "TCP",
  "audit_reconnect_interval": <interval>,
  "audit_reconnect_max_attempts": <max attempts>
}
```

where *<address>* is a string containing the TCP/IP address, *<port>* is a numeric value representing the port, *<interval>* is a numeric value representing the interval in seconds, and *<max attempts>* is a numeric value representing the maximum number of attempts to execute.

Database audits

Once auditing is enabled for your cluster, you can audit individual databases. To do so, use:

- `rladmin`

```
rladmin tune db db:<id|name> db_conns_auditing enabled
```

where the value of the *db:* parameter is either the cluster ID of the database or the database name.

To deactivate auditing, set *db_conns_auditing* to *disabled*.

Use `rladmin info` to retrieve additional details:

```
rladmin info db <id|name>
rladmin info cluster
```

- the [REST API](#)

```
PUT /v1/b dbs/1
{ "db_conns_auditing": true }
```

To deactivate auditing, set db_conns_auditing to false.

You must enable auditing for your cluster before auditing a database; otherwise, an error appears:

Error setting description: Unable to enable DB Connections Auditing before feature configurations are set.
Error setting error_code: db_conns_auditing_config_missing

To resolve this error, enable the protocol for your cluster before attempting to audit a database.

Policy defaults for new databases

To audit connections for new databases by default, use:

- rladmin

```
rladmin tune cluster db_conns_auditing enabled
```

To deactivate this policy, set db_conns_auditing to disabled.

- the [REST API](#)

```
PUT /v1/cluster/policy
{ "db_conns_auditing": true }
```

To deactivate this policy, set db_conns_auditing to false.

Notification examples

Audit event notifications are reported as JSON objects.

New connection

This example reports a new connection for a database:

```
{
  "ts":1655821384,
  "new_conn":
  {
    "id":2285001002 ,
    "srcip":"127.0.0.1",
    "srcp":39338",
    "trgip":"127.0.0.1",
    "trgp":12635",
    "hname":"",
    "bdb_name":DB1",
    "bdb_uid":5"
  }
}
```

Authentication request

Here is a sample authentication request for a database:

```
{
  "ts":1655821384,
  "action":"auth",
  "id":2285001002 ,
  "srcip":"127.0.0.1",
  "srcp":39338,
  "trgip":"127.0.0.1",
  "trgp":12635,
  "hname":"",
  "bdb_name":DB1,
  "bdb_uid":5,
  "status":2,
  "username":user_one,
  "identity":user:1,
  "acl-rules":~* +@all"
}
```

Status reports success or failure. Values of 2 or 8 indicate success; other values indicate failure.

Database disconnect

Here's what's reported when a database connection is closed:

```
{
  "ts":1655821384,
  "close_conn":
  {
    "id":2285001002,
    "srcip":"127.0.0.1",
    "srcp":39338,
    "trgip":"127.0.0.1",
    "trgp":12635,
    "hname":"",
    "bdb_name":DB1,
    "bdb_uid":5
  }
}
```

Notification field reference

The field value that appears immediately after the timestamp describes the action that triggered the notification. The following values may appear:

- new_conn indicates a new external connection
- new_int_conn indicates a new internal connection
- close_conn occurs when a connection is closed
- "action": "auth" indicates an authentication request and can refer to new authentication requests or authorization checks on existing connections

In addition, the following fields may also appear in audit event notifications:

Field name	Description
acl-rules	ACL rules associated with the connection, which includes a rule for the default user.
bdb_name	Destination database name - The name of the database being accessed.
bdb_uid	Destination database ID - The cluster ID of the database being accessed.
hname	Client hostname - The hostname of the client. Currently empty; reserved for future use.
id	Connection ID - Unique connection ID assigned by the proxy.
identity	Identity - A unique ID the proxy assigned to the user for the current connection.
srcip	Source IP address - Source TCP/IP address of the client accessing the Redis database.

Field name	Description
srcp	Source port - Port associated with the source IP address accessing the Redis database. Combine the port with the address to uniquely identify the socket.
status	Status result code - An integer representing the result of an authentication request.
trgip	Target IP address - The IP address of the destination being accessed by the action.
trgp	Target port - The port of the destination being accessed by the action. Combine the port with the destination IP address to uniquely identify the database being accessed.
ts	Timestamp - The date and time of the event, in Coordinated Universal Time (UTC) . Granularity is within one second.
username	Authentication username - Username associated with the connection; can include default for databases that allow default access. (Passwords are not recorded).

Status result codes

The status field reports the results of an authentication request as an integer. Here's what different values mean:

Error value	Error code	Description
0	AUTHENTICATION_FAILED	Invalid username and/or password.
1	AUTHENTICATION_FAILED_TOO_LONG	Username or password are too long.
2	AUTHENTICATION_NOT_REQUIRED	Client tried to authenticate, but authentication isn't necessary.
3	AUTHENTICATION_DIRECTORY_PENDING	Attempting to receive authentication info from the directory in async mode.
4	AUTHENTICATION_DIRECTORY_ERROR	Authentication attempt failed because there was a directory connection error.
5	AUTHENTICATION_SYNCER_IN_PROGRESS	Syncer SASL handshake. Return SASL response and wait for the next request.
6	AUTHENTICATION_SYNCER_FAILED	Syncer SASL handshake. Returned SASL response and closed the connection.
7	AUTHENTICATION_SYNCER_OK	Syncer authenticated. Returned SASL response.
8	AUTHENTICATION_OK	Client successfully authenticated.

Updated: February 13, 2023

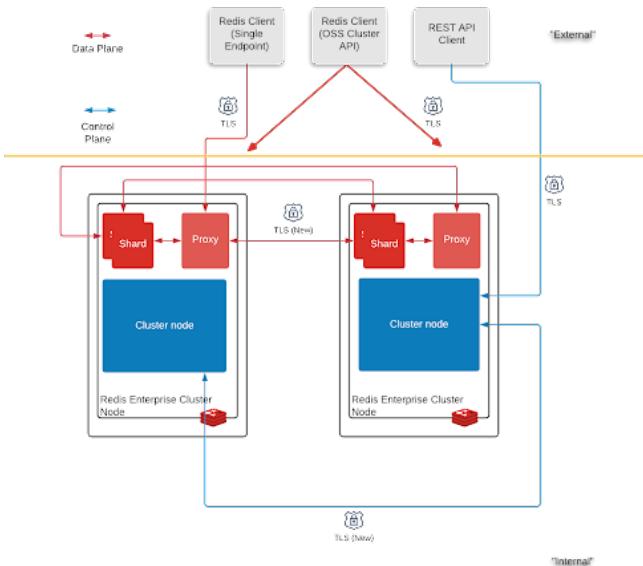
Internode encryption

As of v6.2.4, Redis Enterprise Software supports *internode encryption*, which encrypts internal communication between nodes. This improves the security of data as it travels within a cluster.

Internode encryption is enabled for the *control plane*, which manages the cluster and its databases.

Internode encryption is supported for the *data plane*, which encrypts communication used to replicate shards between nodes and proxy communication with shards located on different nodes.

The following diagram shows how this works.



Data plane encryption is disabled by default for individual databases in order to optimize for performance. Encryption adds latency and overhead; the impact is measurable and varies according to the database, its field types, and the details of the underlying use case.

You can enable data plane encryption for a database by changing the database configuration settings. This lets you choose when to favor performance and when to encrypt data.

Prerequisites

Internode encryption requires certain prerequisites.

You need to:

- Upgrade all nodes in the cluster to v6.2.4 or later.
- Open port 3342 for the TLS channel used for encrypted communication.

Enable data plane encryption

To enable internode encryption for a database (also called *data plane encryption*), you need to enable the appropriate setting for each database you wish to encrypt. To do so, you can:

- Use the admin console to enable the **Internode Encryption** setting from the database **Security** screen.
- Use the `rладmin` command-line utility to set the `data_internode_encryption` setting for the database:

```
rладmin tune db <database_id> data_internode_encryption enable
```

- Use the Redis Enterprise Software REST API to set the `data_internode_encryption` setting for the database.

```
put /v1/bdbs/${database_id}
{ "data_internode_encryption" : true }
```

When you change the data internode encryption setting for a database, all active remote client connections are disconnected. This restarts the internal (DMC) proxy and disconnects all client connections.

To enable data plane encryption by default for new databases, use `rладmin` to tune the cluster:

```
rладmin tune cluster data_internode_encryption enable
```

Encryption ciphers and settings

To encrypt internode communications, Redis Enterprise Software uses TLS 1.2 and the following Cipher suites:

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES128-GCM-SHA256

No configurable settings are exposed; internode encryption is used internally within a cluster and not exposed to any outside service.

Certificate authority and rotation

Starting with v6.2.4, internode communication is managed, in part, by two certificates: one for the control plane and one for the data plane. These certificates are signed by a private certificate authority (CA). The CA is not exposed outside of the cluster, so it cannot be accessed by external processes or services. In addition, each cluster generates a unique CA that is not used anywhere else.

The private CA is generated when a cluster is created or upgraded to 6.2.4.

When nodes join the cluster, the cluster CA is used to generate certificates for the new node, one for each plane. Certificates signed by the private CA are not shared between clusters and they're not exposed outside the cluster.

All certificates signed by the internal CA expire after ninety (90) days and automatically rotate every thirty (30) days. Alerts also monitor certificate expiration and trigger when certificate expiration falls below 45 days. If you receive such an alert, contact support.

You can use the Redis Enterprise Software REST API to rotate certificates manually:

```
POST /v1/cluster/certificates/rotate
```

Updated: August 17, 2023

Admin console security

Redis Enterprise comes with a web-based user interface known as the **admin console**. The admin console provides the following security features:

- Encryption-in-transit using TLS/SSL
- User authentication using LDAP
- Role-based access control

We recommend the following practices:

- **Integrate with an external identity provider:** Redis Enterprise uses [LDAP integration](#) to support external identity providers, such as Active Directory.
- **Implement standard authentication practices:** If your organization does not support LDAP, you can still use Redis Enterprise's [user account security](#). Features include basic password complexity requirements, password expiration, and user login lockouts.
- **Limit session timeouts:** Session timeouts, also known as *automatic sign out*, help prevent unauthorized access. Admin console sessions are allowed to idle for [a period of time](#) before users are required to re-authenticate.

By default, users are signed out after 15 minutes of inactivity. You can set the [timeout period](#).

- **Require HTTPS for API endpoints** - Redis Enterprise comes with a REST API to help automate tasks. This API is available in both an encrypted and unencrypted endpoint for backward compatibility. You can [disable the unencrypted endpoint](#) with no loss in functionality.
- **Configure Transport Layer Security (TLS)** - A common compliance requirement is to [set a minimum version of TLS](#). This helps to make sure that only secure versions of TLS are allowed when accessing the cluster.
- **Install your own certificates** - Redis Enterprise comes with self-signed certificates by default; however, many organizations require that you [use specific CA signed certificates](#).

Updated: June 21, 2022

Encrypt REST API requests

This section details how you can configure encryption for Redis Enterprise Software.

Require HTTPS for API endpoints

By default, the Redis Enterprise Software API supports communication over HTTP and HTTPS. However, you can turn off support for HTTP to ensure that API requests are encrypted.

Before you turn off HTTP support, be sure to migrate any scripts or proxy configurations that use HTTP to the encrypted API endpoint to prevent broken

connections.

To turn off HTTP support for API endpoints, run:

```
rladmin cluster config http_support disabled
```

Updated: June 21, 2022

Certificates

Redis Enterprise Software uses self-signed certificates by default to ensure that the product is secure. If using a self-signed certificate is not the right solution for you, you can import a certificate signed by a certificate authority of your choice.

Here's the list of self-signed certificates that create secure, encrypted connections to your Redis Enterprise cluster:

Certificate name	Description
api	Encrypts REST API requests and responses.
cm	Secures connections to the Redis Enterprise admin console.
metrics_exporter	Sends Redis Enterprise metrics to external monitoring tools over a secure connection.
proxy	Creates secure, encrypted connections between clients and databases.
syncer	For Active-Active or Replica Of databases, encrypts data during the synchronization of participating clusters.

These self-signed certificates are generated on the first node of each Redis Enterprise Software installation and are copied to all other nodes added to the cluster.

When you use the default self-signed certificates and you connect to the admin console over a web browser, you'll see an untrusted connection notification.

Depending on your browser, you can allow the connection for each session or add an exception to trust the certificate for all future sessions.

Updated: December 23, 2022

Create certificates

When you first install Redis Enterprise Software, self-signed certificates are created to encrypt internal traffic. These certificates expire after a year (365 days) and must be renewed.

You can renew these certificates by replacing them with new self-signed certificates or by replacing them with certificates signed by a [certificate authority](#) (CA).

Renew self-signed certificates

As of [v6.2.18-70](#), Redis Enterprise Software includes a script to generate self-signed certificates.

By default, the `generate_self_signed_certs.sh` script is located in `/opt/redislabs/utils/`.

Here, you learn how to use this script to generate new certificates and how to install them.

Step 1: Generate new certificates

Sign in to the machine hosting the cluster's master node and then run the following command:

```
% sudo -u redislabs /opt/redislabs/utils/generate_self_signed_certs.sh \
-f "<DomainName1 DomainName2>" -d <Days> -t <Type>
```

where:

- <DomainName1> is the fully qualified domain name (FQDN) of the cluster. (This is the name given to the cluster when first created.)
- <DomainName2> is an optional FQDN for the cluster. Multiple domain names are allowed, separated by whitespace. Quotation marks (" ") should enclose the full set of names.
- <Days> is an integer specifying the number of days the certificate should be valid. We recommend against setting this longer than a year (365 days).
<Days> is optional and defaults to 365.
- <Type> is a string identifying the name of the certificate to generate.

The following values are supported:

Value	Description
api	The REST API
cm	The admin console
metrics	The metrics exporter
proxy	The database endpoint
syncer	The synchronization process
all	Generates all certificates in a single operation

Type is optional and defaults to all.

When you run the script, it either reports success ("Self signed cert generated successfully") or an error message. Use the error message to troubleshoot any issues.

The following example generates all self signed certificates for mycluster.example.com; these certificates expire one year after the command is run:

```
$ sudo -u redislabs /opt/redislabs/utils/generate_self_signed_certs.sh \
-f "mycluster.example.com"
```

Suppose you want to create an admin console certificate to support two clusters for a period of two years. The following example shows how:

```
$ sudo -u redislabs /opt/redislabs/utils/generate_self_signed_certs.sh \
-f "mycluster.example.com anothercluster.example.com" -d 730 -t cm
```

Here, a certificate file and certificate key are generated to support the following domains:

```
mycluster.example.com
*.mycluster.example.com
anothercluster.example.com
*.anothercluster.example.com
```

Step 2: Locate the new certificate files

When successful, the script generates two .PEM files for each generated certificate: a certificate file and a certificate key, each named after the type of certificate generated (see earlier table for individual certificate names.)

These files can be found in the /tmp directory.

```
$ ls -la /tmp/*.pem
```

Step 3: Set permissions

We recommend setting the permissions of your new certificate files to limit read and write access to the file owner and to set group and other user permissions to read access.

```
$ sudo chmod 644 /tmp/*.pem
```

Step 4: Replace existing certificates

You can use rladm to replace the existing certificates with new certificates:

```
$ rladm cluster certificate set <CertName> certificate_file \  
<CertFilename>.pem key_file <KeyFilename>.pem
```

The following values are supported for the <CertName> parameter:

Value	Description
api	The REST API
cm	The admin console
metrics_exporter	The metrics exporter
proxy	The database endpoint
syncer	The synchronization process

You can also use the REST API. To learn more, see [Update certificates](#).

Create CA-signed certificates

You can use certificates signed by a [certificate authority](#) (CA).

For best results, use the following guidelines to create the certificates.

TLS certificate guidelines

When you create certificates signed by a certificate authority, you need to create server certificates and client certificates. The following provide guidelines that apply to both certificates and guidance for each certificate type.

Guidelines for server and client certificates

1. Include the full [certificate chain](#) when creating certificate .PEM files for either server or client certificates.
2. List (*chain*) certificates in the .PEM file in the following order:

```
-----BEGIN CERTIFICATE-----  
Domain (leaf) certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Intermediate CA certificate  
-----END CERTIFICATE-----  
-----BEGIN CERTIFICATE-----  
Trusted Root CA certificate  
-----END CERTIFICATE-----
```

Server certificate guidelines

Server certificates support clusters.

In addition to the general guidelines described earlier, the following guidelines apply to server certificates:

1. Use the cluster's fully qualified domain name (FQDN) as the certificate Common Name (CN).
2. Set the following values according to the values specified by your security team or certificate authority:
 - Country Name (C)
 - State or Province Name (ST)
 - Locality Name (L)
 - Organization Name (O)
 - Organization Unit (OU)
3. The [Subject Alternative Name](#) (SAN) should include the following values based on the FQDN:

```
dns=<cluster-fqdn>
dns=*.<cluster-fqdn>
dns=internal.<cluster-fqdn>
dns=*.internal.<cluster-fqdn>
```

4. The Extended Key Usage attribute should be set to TLS Web Client Authentication and TLS Web Server Authentication.

5. We strongly recommend using a strong hash algorithm, such as SHA-256 or SHA-512.

Individual operating systems might limit access to specific algorithms. For example, Ubuntu 20.04 limits access to SHA-1. In such cases, Redis Enterprise Software is limited to the features supported by the underlying operating system.

Client certificate guidelines

Client certificates support database connections.

In addition to the general guidelines described earlier, the following guidelines apply to client certificates:

1. The Extended Key Usage attribute should be set to TLS Web Client Authentication.

2. We strongly recommend using a strong hash algorithm, such as SHA-256 or SHA-512.

Individual operating systems might limit access to specific algorithms. For example, Ubuntu 20.04 limits access to SHA-1. In such cases, Redis Enterprise Software is limited to the features supported by the underlying operating system.

Create certificates

The actual process of creating CA-signed certificates varies according to the CA. In addition, your security team may have custom instructions that you need to follow.

Here, we demonstrate the general process using OpenSSL. If your CA provides alternate tools, you should use those according to their instructions.

However you choose to create the certificates, be sure to incorporate the guidelines described earlier.

1. Create a private key.

```
$ openssl genrsa -out <key-file-name>.pem 2048
```

2. Create a certificate signing request.

```
$ openssl req -new -key <key-file-name>.pem -out \
<key-file-name>.csr -config <csr-config-file>.cnf
```

Important: The .CNF file is a configuration file. Check with your security team or certificate authority for help creating a valid configuration file for your environment.

3. Sign the private key using your certificate authority.

The signing process varies for each organization and CA vendor. Consult your security team and certificate authority for specific instructions describing how to sign a certificate.

4. Upload the certificate to your cluster.

You can use [rladmin](#) to replace the existing certificates with new certificates:

```
$ rladm cluster certificate set <CertName> certificate_file \
<CertFilename>.pem key_file <KeyFilename>.pem
```

For a list of values supported by the <CertName> parameter, see the [earlier table](#).

You can also use the REST API. To learn more, see [Update certificates](#).

Updated: April 17, 2023

Update certificates

 | **Warning** - When you update the certificates, the new certificate replaces the same certificates on all nodes in the cluster.

How to update certificates

You can use either the [rladmin](#) command-line interface (CLI) or the [REST API](#) to update certificates.

The new certificates are used the next time the clients connect to the database.

When you upgrade Redis Enterprise Software, the upgrade process copies the certificates that are on the first upgraded node to all of the nodes in the cluster.

 | **Note:** Don't manually overwrite the files located in `/etc/opt/redislabs`. Instead, upload new certificates to a temporary location on one of the cluster nodes, such as the `/tmp` directory.

Use the CLI

To replace certificates with the [rladmin](#) CLI, run the `cluster certificate set` command:

```
rladmin cluster certificate set <cert-name> certificate_file <cert-file-name>.pem key_file <key-file-name>.pem
```

Replace the following variables with your own values:

- `<cert-name>` - The name of the certificate you want to replace. See the [certificates table](#) for the list of valid certificate names.
- `<cert-file-name>` - The name of your certificate file
- `<key-file-name>` - The name of your key file

For example, to replace the admin console (`cm`) certificate with the private key `key.pem` and the certificate file `cluster.pem`:

```
rladmin cluster certificate set cm certificate_file cluster.pem key_file key.pem
```

Use the REST API

To replace a certificate using the REST API, use [`PUT /v1/cluster/update_cert`](#):

```
PUT https://[host][:port]/v1/cluster/update_cert
  { "name": "<cert_name>", "key": "<key>", "certificate": "<cert>" }
```

Replace the following variables with your own values:

- `<cert_name>` - The name of the certificate to replace. See the [certificates table](#) for the list of valid certificate names.
- `<key>` - The contents of the `*_key.pem` file

 | **Tip** - The key file contains \n end of line characters (EOL) that you cannot paste into the API call. You can use sed `-z 's/\n/\\n/g'` to escape the EOL characters.

- `<cert>` - The contents of the `*_cert.pem` file

Replica Of database certificates

This section describes how to update certificates for Replica Of (also known as Active-Passive) databases.

Update proxy certificates

To update the proxy certificate on clusters running Replica Of (Active-Passive) databases:

- **Step 1:** Use [rladmin](#) or the REST API to update the proxy certificate on the source database cluster.
- **Step 2:** From the admin console, update the destination database (`replica`) configuration with the [new certificate](#).



Note:

- Perform Step 2 as quickly as possible after performing Step 1. Connections using the previous certificate are rejected after applying the new certificate. Until both steps are performed, recovery of the database sync cannot be established.

Active-Active database certificates

Update proxy certificates

To update proxy certificate on clusters running Active-Active databases:

- **Step 1:** Use `radmin` or the REST API to update proxy certificates on a single cluster, multiple clusters, or all participating clusters.
- **Step 2:** Use the `crdb-cli` utility to update Active-Active database configuration from the command line. Run the following command once for each Active-Active database residing on the modified clusters:

```
crdb-cli crdb update --crdb-guid <CRDB-GUID> --force
```



Note:

- Perform Step 2 as quickly as possible after performing Step 1. Connections using the previous certificate are rejected after applying the new certificate. Until both steps are performed, recovery of the database sync cannot be established.
- Do not run any other `crdb-cli crdb update` operations between the two steps.

Update syncer certificates

To update your syncer certificate on clusters running Active-Active databases, follow these steps:

- **Step 1:** Update your syncer certificate on one or more of the participating clusters using the `radmin` command, REST API, or admin console. You can update a single cluster, multiple clusters, or all participating clusters.
- **Step 2:** Update the Active-Active database configuration from the command line with the `crdb-cli` utility. Run this command once for each Active-Active database that resides on the modified clusters:

```
crdb-cli crdb update --crdb-guid <CRDB-GUID> --force
```



Note:

- Run step 2 as quickly as possible after step 1. Between the two steps, new syncer connections that use the 'old' certificate will get rejected by the cluster that has been updated with the new certificate (in step 1).
- Do not run any other `crdb-cli crdb update` operations between the two steps.
- **Known limitation:** Updating syncer certificate on versions prior to 6.0.20-81 will restart the proxy and syncer connections. In these cases, we recommend scheduling certificate replacement carefully to minimize customer impact.

Updated: August 17, 2023

Enable OCSP stapling

OCSP ([Online Certificate Status Protocol](#)) lets a client or server verify the status (GOOD, REVOKED, or UNKNOWN) of a certificate maintained by a third-party certificate authority (CA).

To check whether a certificate is still valid or has been revoked, a client or server can send a request to the CA's OCSP server (also called an OCSP responder). The OCSP responder checks the certificate's status in the CA's [certificate revocation list](#) and sends the status back as a signed and timestamped response.

OCSP stapling overview

With OCSP enabled, the Redis Enterprise server regularly polls the CA's OCSP responder for the certificate's status. After it receives the response, the

server caches this status until its next polling attempt.

When a client tries to connect to the Redis Enterprise server, they perform a [TLS handshake](#) to authenticate the server and create a secure, encrypted connection. During the TLS handshake, [OCSP stapling](#) lets the Redis Enterprise server send (or “staple”) the cached certificate status to the client.

If the stapled OCSP response confirms the certificate is still valid, the TLS handshake succeeds and the client connects to the server.

The TLS handshake fails and the client blocks the connection to the server if the stapled OCSP response indicates either:

- The certificate has been revoked.
- The certificate’s status is unknown. This can happen if the OCSP responder fails to send a response.

Set up OCSP stapling

You can configure and enable OCSP stapling for your Redis Enterprise cluster with the [admin console](#), the [REST API](#), or [rladmin](#).

While OCSP is enabled, the server always staples the cached OCSP status when a client tries to connect. It is the client’s responsibility to use the stapled OCSP status. Some Redis clients, such as [Jedis](#) and [redis-py](#), already support OCSP stapling, but others might require additional configuration.

Admin console method

To set up OCSP stapling with the Redis Enterprise admin console:

1. Go to **Cluster > Security > OCSP**.
2. In the **Responder URI** section, select **Replace Certificate** to update the proxy certificate.
3. Provide the key and certificate signed by your third-party CA, then select **Save**.
4. Configure query settings if you don’t want to use their default values:

Name	Default value	Description
Query frequency	1 hour	The time interval between OCSP queries to the responder URI.
Response timeout	1 second	The time interval in seconds to wait for a response before timing out.
Recovery frequency	1 minute	The time interval between retries after a failed query.
Recovery maximum tries	5	The number of retries before the validation query fails and invalidates the certificate.

5. Select **Enable** to turn on OCSP stapling.

REST API method

To set up OCSP stapling with the [REST API](#):

1. Use the [REST API](#) to replace the proxy certificate with a certificate signed by your third-party CA.
2. To configure and enable OCSP, send a [PUT request to the /v1/ocsp endpoint](#) and include an [OCSP JSON object](#) in the request body:

```
{  
  "ocsp_functionality": true,  
  "query_frequency": 3600,  
  "response_timeout": 1,  
  "recovery_frequency": 60,  
  "recovery_max_tries": 5  
}
```

rladmin method

To set up OCSP stapling with the [rladmin](#) command-line utility:

1. Use [rladmin](#) to replace the proxy certificate with a certificate signed by your third-party CA.
2. Update the cluster’s OCSP settings with the [rladmin cluster ocsp config](#) command if you don’t want to use their default values.

For example:

```
rladmin cluster ocsp config recovery_frequency set 30
```

3. Enable OCSP:

```
rladmin cluster ocsp config ocsp_functionality set enabled
```

Updated: August 17, 2023

Transport Layer Security (TLS)

[Transport Layer Security \(TLS\)](#), a successor to SSL, ensures the privacy of data sent between applications and Redis databases. TLS also secures connections between Redis Enterprise Software nodes.

You can use [TLS authentication](#) for the following types of communication:

- Communication from clients (applications) to your database
- Communication from your database to other clusters for replication using [Replica Of](#)
- Communication to and from your database to other clusters for synchronization using [Active-Active](#)

Protocols and ciphers

TLS protocols and ciphers define the overall suite of algorithms that clients are able to connect to the servers with.

You can change the [TLS protocols](#) and [ciphers](#) to improve the security of your Redis Enterprise cluster and databases. The default settings are in line with industry best practices, but you can customize them to match the security policy of your organization.

Client-side encryption

Client-side encryption may be used to help encrypt data through its lifecycle. This comes with some limitations. Operations that must operate on the data, such as increments, comparisons, and searches will not function properly. Client-side encryption is used to help protect data in use.

You can write client-side encryption logic directly in your own application or use functions built into clients such as the Java Lettuce cipher codec.

Updated: June 6, 2022

Enable TLS

You can use TLS authentication for one or more of the following types of communication:

- Communication from clients (applications) to your database
- Communication from your database to other clusters for replication using [Replica Of](#)
- Communication to and from your database to other clusters for synchronization using [Active-Active](#)

Enable TLS for client connections

To enable TLS for client connections:

1. From your database's **Security** tab, select **Edit**.
2. In the **Secure connections (via TLS - Transport Layer Security)** section, select **Mutual TLS authentication**.

Secure connections (via TLS - Transport Layer Security)



Off

Server authentication only

Mutual TLS authentication

Communication scope



Replica Of source database and clients



Mutual TLS (Client authentication)

Add syncer certificate from target and clients certificate



+ Add certificate

Additional certificate validations

No validation



+ Add validation

3. For Communication scope, select Replica Of source database and clients.

4. For each client certificate, select + Add certificate, paste or upload the client certificate, then select Done.

If your database uses Replica Of or Active-Active replication, you also need to add the syncer certificates for the participating clusters. See [Enable TLS for Replica Of cluster connections](#) or [Enable TLS for Active-Active cluster connections](#) for instructions.

5. You can configure Additional certificate validations to further limit connections to clients with valid certificates.

Additional certificate validations occur only when loading a [certificate chain](#) that includes the [root certificate](#) and intermediate [CA](#) certificate but does not include a leaf (end-entity) certificate. If you include a leaf certificate, mutual client authentication skips any additional certificate validations.

1. Select a certificate validation option.

Validation option	Description
No validation	Authenticates clients with valid certificates. No additional validations are enforced.
By Subject Alternative Name	A client certificate is valid only if its Common Name (CN) matches an entry in the list of valid subjects. Ignores other Subject attributes.
By full Subject Name	A client certificate is valid only if its Subject attributes match an entry in the list of valid subjects.

2. If you selected **No validation**, you can skip this step. Otherwise, select + Add validation to create a new entry and then enter valid [Subject](#) attributes for your client certificates. All [Subject](#) attributes are case-sensitive.

Subject attribute (case-sensitive)	Description
Common Name (CN)	Name of the client authenticated by the certificate (required)
Organization (O)	The client's organization or company name
Organizational Unit (OU)	Name of the unit or department within the organization
Locality (L)	The organization's city
State / Province (ST)	The organization's state or province
Country (C)	2-letter code that represents the organization's country

You can only enter a single value for each field, except for the *Organizational Unit (OU)* field. If your client certificate has a *Subject* with multiple *Organizational Unit (OU)* values, enter a comma-separated list of values with each value enclosed in quotation marks (for example "unit1", "unit2").

Breaking change: If you use the [REST API](#) instead of the admin console to configure additional certificate validations, note that `authorized_names` is deprecated as of Redis Enterprise v6.4.2. Use `authorized_subjects` instead. See the [BDB object reference](#) for more details.

6. Select **Save**.

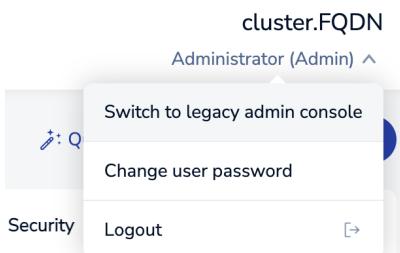
7. (Optional) By default, Redis Enterprise Software validates client certificate expiration dates. You can use `radmin` to turn off this behavior.

```
radmin tune db < db:id | name > mtls_allow_outdated_certs enabled
```

Enable TLS for Active-Active cluster connections

To enable TLS for Active-Active cluster connections:

1. If you are using the new admin console, switch to the legacy admin console.



2. [Retrieve syncer certificates](#).

3. [Configure TLS certificates for Active-Active](#).

4. [Configure TLS on all participating clusters](#).



Note: You cannot enable or turn off TLS after the Active-Active database is created, but you can change the TLS configuration.

Retrieve syncer certificates

For each participating cluster, copy the syncer certificate from the [general settings tab](#).

settings

general redis modules LDAP alerts

Cluster key [\(i\)](#)

Current key details

Customer identifier: N/A

Cluster name: N/A

Max number of shards: 4

Key expiration date: 4/15/2021 1:49:14 PM

Proxy Certificate

```
-----BEGIN CERTIFICATE-----  
MIIDbzCCAlgAwIBAgIJALFrSDlnu/vvMA0GCSqGSIb3DQEBCwUAMFAXFTATBqNV  
BAMMDD1iMDUyNWiyZmY4YjEQMA4GA1UECwHcmVkaXNkYjE1MCNGA1UECgwcUmVk  
aNMYWJzIEVudGVycHJpc2UgQ2x1c3RlcjaeFw0yMTAzMTYxMzQ5MT2aFw0yMjAz  
MTYxMzQ5MTZaMFAxFTATBqNVBAMMDD1iMDUyNWiyZmY4YjEQMA4GA1UECwHcmVk  
aNkYjE1MCNGA1UECgwcUmVkaXNMYWJzIEVudGVycHJpc2UgQ2x1c3RlcjCCASIw  
DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKmCYdtpT561Cxmp6VDEpfyBjlka  
J6InCghGN0xxmXds4bndIP8f1dAGhvQkaReRAO+1TL6W81TVPCrnF/VVNnGQA8kW  
-----END CERTIFICATE-----
```

Syncer Certificate

```
-----BEGIN CERTIFICATE-----  
MIIDbTCALgAwIBAgIJAM3gPamlK94tMA0GCSqGSIb3DQEBCwUAME8xFTATBqNV  
BAMMDD1iMDUyNWiyZmY4YjEPMA0GA1UECwHc3luY2VyMSUwIwYDVQQKDBxSZWRp  
c0xhYnMgRW50ZXJwcm1zZSBdbHVzdGVyMB4XDThxMDMxNjEzNDkxNv0xDThyMDMx  
NjEzNDkxNv0wTzEVMBMGA1UEAwMOWIwNTI1YjJmZjhIMQ8wDQYDVQQLDAzeW5j  
ZXIxJTAjBgvNBAb0MHFJlZGlzTGFicyBFbnRlcnByaXN1IENsdXN0ZXiwggiMA0G  
CSqGSIb3DOEBAOUAA4IBDwAwaEKoIBAOtce53B5OvDRU7D0S20xuCDQG1oB0u0  
-----END CERTIFICATE-----
```

Configure TLS certificates for Active-Active

1. During database creation (see [Create an Active-Active Geo-Replicated Database](#), select **Edit** from the configuration tab.
2. Enable **TLS**.

- **Enforce client authentication** is selected by default. If you clear this option, you will still enforce encryption, but TLS client authentication will be deactivated.

3. Select **Require TLS for CRDB communication only** from the dropdown menu.



4. Select **Add**

5. Paste a syncer certificate into the text box.



6. Save the syncer certificate.
7. Repeat this process, adding the syncer certificate for each participating cluster.
8. Optional: If also you want to require TLS for client connections, select **Require TLS for All Communications** from the dropdown and add client certificates as well.
9. Select **Update** at the bottom of the screen to save your configuration.

Configure TLS on all participating clusters

Repeat this process on all participating clusters.

To enforce TLS authentication, Active-Active databases require syncer certificates for each cluster connection. If every participating cluster doesn't have a

syncer certificate for every other participating cluster, synchronization will fail.

Enable TLS for Replica Of cluster connections

To enable TLS for Replica Of cluster connections:

1. For each cluster hosting a replica:
 1. Go to Cluster > Security > Certificates.
 2. Expand the **Replica Of and Active-Active authentication (Syncer certificate)** section.

Replica Of and Active-Active authentication (Syncer certificate) ^

Subject name:	Issuer:	Valid from:	Expiration date:	i
cluster.FQDN	cluster.FQDN	20 Jul 2023 5:32 AM	19 Jul 2024 5:32 AM	

-----BEGIN CERTIFICATE-----

-----END CERTIFICATE-----

[Replace Certificate](#) [Download](#) [Copy](#)

3. Download or copy the syncer certificate.
2. From the **Security** tab of the Replica Of source database, select **Edit**.
3. In the **Secure connections (via TLS - Transport Layer Security)** section, select **Mutual TLS authentication**.

Secure connections (via TLS - Transport Layer Security)



Off

Server authentication only

Mutual TLS authentication

Communication scope



Replica Of source database



Mutual TLS (Client authentication)

Add syncer certificate from target



+ Add certificate

Additional certificate validations

No validation



+ Add validation

4. For Communication scope, select Replica Of source database.

5. Select + Add certificate, paste or upload the syncer certificate, then select Done.

Repeat this process, adding the syncer certificate for each cluster hosting a replica of this database.

6. (Optional) If you also want to require TLS for client connections, change Communication scope to Replica Of source database and clients and add client certificates.

7. Select Save.

Updated: August 17, 2023

Configure TLS protocol

You can change TLS protocols to improve the security of your Redis Enterprise cluster and databases. The default settings are in line with industry best practices, but you can customize them to match the security policy of your organization.

Configure TLS protocol

The communications for which you can modify TLS protocols are:

- Control plane - The TLS configuration for cluster administration.

- Data plane - The TLS configuration for the communication between applications and databases.
- Discovery service (Sentinel) - The TLS configuration for the [discovery service](#).

You can configure the TLS protocols with the `rladmin` commands shown here or with the REST API.

 **Warning** - After you set the minimum TLS version, Redis Enterprise Software does not accept communications with TLS versions older than the specified version.

TLS support depends on the operating system. You cannot enable support for protocols or versions that aren't supported by the operating system running Redis Enterprise Software. In addition, updates to the operating system or to Redis Enterprise Software can impact protocol and version support.

To illustrate, version 6.2.8 of Redis Enterprise Software removed support for TLS 1.0 and TLS 1.1 on Red Hat Enterprise Linux 8 (RHEL 8) because that operating system [does not enable support](#) for these versions by default.

If you have trouble enabling specific versions of TLS, verify that they're supported by your operating system and that they're configured correctly.

 **Note:** TLSv1.2 is generally recommended as the minimum TLS version for encrypted communications. Check with your security team to confirm which TLS protocols meet your organization's policies.

Control plane

To set the minimum TLS protocol for the control plane:

- Default minimum TLS protocol: TLSv1.2
- Syntax: `rladmin cluster config min_control_TLS_version <TLS_Version>`
- TLS versions available:
 - For TLSv1 - 1
 - For TLSv1.1 - 1.1
 - For TLSv1.2 - 1.2

For example:

```
rladmin cluster config min_control_TLS_version 1.2
```

Data plane

To set the minimum TLS protocol for the data path:

- Default minimum TLS protocol: TLSv1.2
- Syntax: `rladmin cluster config min_data_TLS_version <TLS_Version>`
- TLS versions available:
 - For TLSv1 - 1
 - For TLSv1.1 - 1.1
 - For TLSv1.2 - 1.2

For example:

```
rladmin cluster config min_data_TLS_version 1.2
```

Discovery service

To enable TLS for the discovery service:

- Default: Allows both TLS and non-TLS connections
- Syntax: `rladmin cluster config sentinel_ssl_policy <ssl_policy>`
- `ssl_policy` values available:
 - `allowed` - Allows both TLS and non-TLS connections
 - `required` - Allows only TLS connections
 - `disabled` - Allows only non-TLS connections

To set the minimum TLS protocol for the discovery service:

- Default minimum TLS protocol: TLSv1.2
- Syntax: `rladmin cluster config min_sentinel_TLS_version <TLS_Version>`
- TLS versions available:
 - For TLSv1 - 1
 - For TLSv1.1 - 1.1

- For TLSv1.2 - 1.2

To enforce a minimum TLS version for the discovery service, run the following commands:

1. Allow only TLS connections:

```
rladmin cluster config sentinel_ssl_policy required
```

2. Set the minimal TLS version:

```
rladmin cluster config min_sentinel_TLS_version 1.2
```

3. Restart the discovery service on all cluster nodes to apply your changes:

```
supervisorctl restart sentinel_service
```

Updated: August 17, 2023

Configure cipher suites

Ciphers are algorithms that help secure connections between clients and servers. You can change the ciphers to improve the security of your Redis Enterprise cluster and databases. The default settings are in line with industry best practices, but you can customize them to match the security policy of your organization.

Configure cipher suites

The communications for which you can modify ciphers are:

- Control plane - The TLS configuration for cluster administration.
- Data plane - The TLS configuration for the communication between applications and databases.
- Discovery service (Sentinel) - The TLS configuration for the [discovery service](#).

You can configure ciphers with the [rladmin](#) commands shown here or with the [REST API](#).



Warning - Configuring cipher suites overwrites existing ciphers rather than appending new ciphers to the list.

When you modify your cipher suites, make sure:

- The configured TLS version matches the required cipher suites.
- The certificates in use are properly signed to support the required cipher suites.



Note:

- Redis Enterprise Software doesn't support static [Diffie–Hellman \(DH\)](#) key exchange ciphers.
- It does support Ephemeral Diffie–Hellman (DHE or ECDHE) key exchange ciphers on Red Hat Enterprise Linux (RHEL) 8 and Bionic OS.

Control plane

As of Redis Enterprise Software version 6.0.12, control plane cipher suites can use the BoringSSL library format for TLS connections to the admin console. See the BoringSSL documentation for a full list of available [BoringSSL configurations](#).

To configure the cipher suites for cluster communication, use the following [rladmin](#) command syntax:

```
rladmin cluster config cipher_suites <BoringSSL cipher list>
```

See the example below to configure cipher suites for the control plane:

```
rладmin cluster config cipher_suites ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:!3DES
```



Note:

- The phrase '!3DES' disables all 3DES cipher suites.

Data plane

Data plane cipher suites use the OpenSSL library format in Redis Enterprise Software version 6.0.20 or later. For a list of available OpenSSL configurations, see [Ciphers \(OpenSSL\)](#).

To configure the cipher suites for communications between applications and databases, use the following `rладmin` command syntax:

```
rладmin cluster config data_cipher_list <OpenSSL cipher list>
```

See the example below to configure cipher suites for the data plane:

```
rладmin cluster config data_cipher_list AES128-SHA:AES256-SHA:!3DES
```



Note:

- The phrase '!3DES' disables all 3DES cipher suites.

Discovery service

Sentinel service cipher suites use the golang.org OpenSSL format for [discovery service](#) TLS connections in Redis Enterprise Software version 6.0.20 or later. See their documentation for a list of [available configurations](#).

To configure the discovery service cipher suites, use the following `rладmin` command syntax:

```
rладmin cluster config sentinel_cipher_suites <golang cipher list>
```

See the example below to configure cipher suites for the sentinel service:

```
rладmin cluster config sentinel_cipher_suites  
TLS_RSA_WITH_AES_128_CBC_SHA:TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
```

Updated: August 17, 2023

Reference

[Redis clients](#)

Redis client libraries allow you to connect to Redis instances from within your application. This section provides an overview of several recommended Redis clients for popular programming and scripting languages.

For a list of available Redis clients sorted by language, see [Clients \(redis.io\)](#).

[Command-line utilities](#)

Reference for Redis Enterprise Software command-line utilities, including `rладmin`, `redis-cli`, `crdb-cli`, and `rlcheck`.

[REST API](#)

Documents the REST API available to Redis Enterprise Software deployments.

Compatibility with open source Redis

Documents compatibility of open source Redis commands and configuration settings with Redis Enterprise Software and Redis Cloud.

Terminology

Explains terms used in Redis Enterprise Software documentation.

Updated: December 21, 2022

Develop with Redis clients

To connect to Redis instances from within your application, use a Redis client library that matches your application's language.

Official clients

Language	Client name
.Net	NRedisStack
Go	go-redis
Java	Jedis
Node.js	node-redis
Python	redis-py

Select a client name to see its quick start.

Other clients

For a list of community-driven Redis clients, which are available for more programming languages, see [Other Clients](#).

Updated: August 17, 2023

Benchmark a Auto Tiering enabled database

Auto Tiering on Redis Enterprise Software lets you use cost-effective Flash memory as a RAM extension for your database.

But what does the performance look like as compared to a memory-only database, one stored solely in RAM?

These scenarios use the `memtier_benchmark` utility to evaluate the performance of a Redis Enterprise Software deployment, including the trial version.

The `memtier_benchmark` utility is located in `/opt/redislabs/bin/` of Redis Enterprise Software deployments. To test performance for cloud provider deployments, see the [memtier-benchmark GitHub project](#).

For additional, such as assistance with larger clusters, [contact support](#).

Benchmark and performance test considerations

These tests assume you're using a trial version of Redis Enterprise Software and want to test the performance of a Auto Tiering enabled database in the following scenarios:

- Without replication: Four (4) master shards
- With replication: Two (2) primary and two replica shards

With the trial version of Redis Enterprise Software you can create a cluster of up to four shards using a combination of database configurations, including:

- Four databases, each with a single master shard
- Two highly available databases with replication enabled (each database has one master shard and one replica shard)
- One non-replicated clustered database with four master shards
- One highly available and clustered database with two master shards and two replica shards

Test environment and cluster setup

For the test environment, you need to:

1. Create a cluster with three nodes.
2. Prepare the flash memory.
3. Configure the load generation tool.

Creating a three-node cluster

This performance test requires a three-node cluster.

You can run all of these tests on Amazon AWS with these hosts:

- 2 x i3.2xlarge (8 vCPU, 61 GiB RAM, up to 10GBit, 1.9TB NVMe SSD)

These nodes serve RoF data

- 1 x m4.large, which acts as a quorum node

To learn how to install Redis Enterprise Software and set up a cluster, see:

- [Redis Enterprise Software quickstart](#) for a test installation
- [Install and upgrade](#) for a production installation

These tests use a quorum node to reduce AWS EC2 instance use while maintaining the three nodes required to support a quorum node in case of node failure. Quorum nodes can be on less powerful instances because they do not have shards or support traffic.

As of this writing, i3.2xlarge instances are required because they support NVMe SSDs, which are required to support RoF. Auto Tiering requires Flash-enabled storage, such as NVMe SSDs.

For best results, compare performance of a Flash-enabled deployment to the performance in a RAM-only environment, such as a strictly on-premises deployment.

Prepare the flash memory

After you install RS on the nodes, the flash memory attached to the i3.2xlarge instances must be prepared and formatted with the `/opt/redislabs/sbin/prepare_flash.sh` script.

Set up the load generation tool

The memtier_benchmark load generator tool generates the load on the RoF databases. To use this tool, install RS on a dedicated instance that is not part of the RS cluster but is in the same region/zone/subnet of your cluster. We recommend that you use a relatively powerful instance to avoid bottlenecks at the load generation tool itself.

For these tests, the load generation host uses a c4.8xlarge instance type.

Database configuration parameters

Create a Auto Tiering test database

You can use the RS admin console to create a test database. We recommend that you use a separate database for each test case with these requirements:

Parameter	With replication	Without replication	Description
Name	test-1	test-2	The name of the test database
Memory limit	100 GB	100 GB	The memory limit refers to RAM+Flash, aggregated across all the shards of the database, including master and replica shards.

Parameter	With replication	Without replication	Description
RAM limit	0.3	0.3	RoF always keeps the Redis keys and Redis dictionary in RAM and additional RAM is required for storing hot values. For the purpose of these tests 30% RAM was calculated as an optimal value.
Replication	Enabled	Disabled	A database with no replication has only master shards. A database with replication has master and replica shards.
Data persistence	None	None	No data persistence is needed for these tests.
Database clustering	Enabled	Enabled	A clustered database consists of multiple shards.
Number of (master) shards	2	4	Shards are distributed as follows: - With replication: One master shard and one replica shard on each node - Without replication: Two master shards on each node
Other parameters	Default	Default	Keep the default values for the other configuration parameters.

Data population

Populate the benchmark dataset

The memtier_benchmark load generation tool populates the database. To populate the database with N items of 500 Bytes each in size, on the load generation instance run:

```
$ memtier_benchmark -s $DB_HOST -p $DB_PORT --hide-histogram
--key-maximum=$N -n allkeys -d 500 --key-pattern=P:P --ratio=1:0
```

Set up a test database:

Parameter	Description
Database host (-s)	The fully qualified name of the endpoint or the IP shown in the RS database configuration
Database port (-p)	The endpoint port shown in your database configuration
Number of items (-key-maximum)	With replication: 75 Million Without replication: 150 Million
Item size (-d)	500 Bytes

Centralize the keyspace

With replication

To create roughly 20.5 million items in RAM for your highly available clustered database with 75 million items, run:

```
$ memtier_benchmark -s $DB_HOST -p $DB_PORT --hide-histogram
--key-minimum=27250000 --key-maximum=47750000 -n allkeys
--key-pattern=P:P --ratio=0:1
```

To verify the database values, use **Values in RAM** metric, which is available from the **Metrics** tab of your database in the admin console.

Without replication

To create 41 million items in RAM without replication enabled and 150 million items, run:

```
$ memtier_benchmark -s $DB_HOST -p $DB_PORT --hide-histogram  
--key-minimum=54500000 --key-maximum=95500000 -n allkeys  
--key-pattern=P:P --ratio=0:1
```

Test runs

Generate load

With replication

We recommend that you do a dry run and double check the RAM Hit Ratio on the **metrics** screen in the RS admin console before you write down the test results.

To test RoF with an 85% RAM Hit Ratio, run:

```
$ memtier_benchmark -s $DB_HOST -p $DB_PORT --pipeline=11 -c 20 -t 1  
-d 500 --key-maximum=75000000 --key-pattern=G:G --key-stddev=5125000  
--ratio=1:1 --distinct-client-seed --randomize --test-time=600  
--run-count=1 --out-file=test.out
```

Without replication

Here is the command for 150 million items:

```
$ memtier_benchmark -s $DB_HOST -p $DB_PORT --pipeline=24 -c 20 -t 1  
-d 500 --key-maximum=150000000 --key-pattern=G:G --key-stddev=10250000  
--ratio=1:1 --distinct-client-seed --randomize --test-time=600  
--run-count=1 --out-file=test.out
```

Where:

Parameter	Description
Access pattern (-key-pattern) and standard deviation (-key-stddev)	Controls the RAM Hit ratio after the centralization process is complete
Number of threads (-t and -c)\	Controls how many connections are opened to the database, whereby the number of connections is the number of threads multiplied by the number of connections per thread (-t) and number of clients per thread (-c)
Pipelining (-pipeline)\	Pipelining allows you to send multiple requests without waiting for each individual response (-t) and number of clients per thread (-c)
Read\write ratio (-ratio)\	A value of 1:1 means that you have the same number of write operations as read operations (-t) and number of clients per thread (-c)

Test results

Monitor the test results

You can either monitor the results in the **metrics** tab of the admin console or with the **memtier_benchmark** output. However, be aware that:

- The **memtier_benchmark** results include the network latency between the load generator instance and the cluster instances.
- The metrics shown in the admin console do *not* include network latency.

Expected results

You should expect to see an average throughput of:

- Around 160,000 ops/sec when testing without replication (i.e. Four master shards)
- Around 115,000 ops/sec when testing with enabled replication (i.e. Four master and 2 replica shards)

In both cases, the average latency should be below one millisecond.

Command-line utilities

Redis Enterprise Software includes a set of utilities to help you manage and test your cluster. To use a utility, run it from the command line.

Public utilities

Administrators can use these CLI tools to manage and test a Redis Enterprise cluster. You can find the binaries in the `/opt/redislabs/bin/` directory.

Utility	Description
<code>crdb-cli</code> (manage Active-Active)	Manage Active-Active databases.
<code>redis-cli</code> (run Redis commands)	Run Redis commands.
<code>rladmin</code> (manage cluster)	Manage Redis Enterprise clusters and databases.
<code>rlcheck</code> (verify nodes)	Verify nodes.

Internal utilities

The `/opt/redislabs/bin/` directory also contains utilities used internally by Redis Enterprise Software and for troubleshooting.

 | **Warning** - Do not use these tools for normal operations.

Utility	Description
<code>bdb-cli</code>	<code>redis-cli</code> connected to a database.
<code>ccs-cli</code>	Inspect Cluster Configuration Store.
<code>cnm-ctl</code>	Manages services for provisioning, migration, monitoring, resharding, rebalancing, deprovisioning, and autoscaling.
<code>consistency_checker</code>	Checks the consistency of Redis instances.
<code>crdbtop</code>	Monitor Active-Active databases.
<code>debug_mode</code>	Enables debug mode.
<code>debuginfo</code>	Collects cluster information.
<code>dmc-ctl</code>	Configure and monitor the DMC proxy.
<code>pdns_control</code>	Sends commands to a running PowerDNS nameserver.
<code>redis_ctl</code>	Stops or starts Redis instances.
<code>rl_rdbloader</code>	Load RDB backup files to a server.
<code>rlutil</code>	Maintenance utility.
<code>shard-cli</code>	<code>redis-cli</code> connected to a shard.
<code>supervisorctl</code>	Manages the lifecycles of Redis Enterprise services.

Updated: July 7, 2022

crdb-cli

An [Active-Active database](#) (also known as CRDB or conflict-free replicated database) replicates your data across Redis Enterprise Software clusters located in geographically distributed regions. Active-Active databases allow read-write access in all locations, making them ideal for distributed applications that require fast response times and disaster recovery.

The Active-Active database on an individual cluster is called an **instance**. Each cluster that hosts an instance is called a **participating cluster**.

An Active-Active database requires two or more participating clusters. Each instance is responsible for updating the instances that reside on other participating clusters with the transactions it receives. Write conflicts are resolved using [conflict-free replicated data types](#) (CRDTs).

To programmatically maintain an Active-Active database and its instances, you can use the `crdb-ctl` command-line tool.

crdb-cli commands

Command	Description
crdb	Manage Active-Active databases.
task	Manage Active-Active tasks.

Use the crdb-cli

To use the `crdb-cli` tool, use SSH to sign in to a Redis Enterprise host with a user that belongs to the group that Redis Enterprise Software was installed with (Default: `redislabs`). If you sign in with a non-root user, you must add `/opt/redislabs/bin` to your PATH environment variables.

`crdb-cli` commands use the syntax: `crdb-cli <command> <arguments>` to let you:

- Create, list, update, flush, or delete an Active-Active database.
- Add or remove an instance of the Active-Active database on a specific cluster.

Each command creates a task.

By default, the command runs immediately and displays the result in the output.

If you use the `--no-wait` flag, the command runs in the background so that your application is not delayed by the response.

Use the [crdb-cli task commands](#) to manage Active-Active database tasks.

For each `crdb-cli` command, you can use `--help` for additional information about the command.

Updated: October 13, 2022

crdb-cli crdb commands

Use `crdb-cli crdb` commands to manage Active-Active databases.

crdb-cli crdb commands

Command	Description
add-instance	Adds a peer replica to an Active-Active database.
create	Creates an Active-Active database.
delete	Deletes an Active-Active database.
flush	Clears all keys from an Active-Active database.
get	Shows the current configuration of an Active-Active database.
health-report	Shows the health report of an Active-Active database.
list	Shows a list of all Active-Active databases.
purge-instance	Deletes data from a local instance and removes it from the Active-Active database.
remove-instance	Removes a peer replica from an Active-Active database.
update	Updates the configuration of an Active-Active database.

Updated: July 7, 2022

crdb-cli crdb add-instance

Adds a peer replica to an existing Active-Active database in order to host the database on another cluster. This creates an additional active instance of the database on the specified cluster.

```
crdb-cli crdb add-instance --crdb-guid <guid>
    --instance
  fqdn=<cluster_fqdn>,username=<username>,password=<password>[,url=<url>,replication_endpoint=<endpoint>]
    [ --compression <0-6> ]
    [ --no-wait ]
```

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)
instance	strings	The connection information for the new participating cluster (required)
compression	0-6	The level of data compression: 0=Compression disabled 6=High compression and resource load (Default: 3)
no-wait		Does not wait for the task to complete

Returns

Returns the task ID of the task that is adding the new instance.

If `--no-wait` is specified, the command exits. Otherwise, it will wait for the instance to be added and return `finished`.

Example

```
$ crdb-cli crdb add-instance --crdb-guid db6365b5-8aca-4055-95d8-7eb0105c0b35 \
    --instance fqdn=cluster2.redis.local,username=admin@redis.local,password=admin-password
Task f809fae7-8e26-4c8f-9955-b74dbbd47949 created
---> Status changed: queued -> started
---> Status changed: started -> finished
```

Updated: July 7, 2022

crdb-cli crdb create

Creates an Active-Active database.

```

crdb-cli crdb create --name <name>
  --memory-size <maximum_memory>
  --instance
fqdn=<cluster_fqdn>,username=<username>,password=<password>[,url=<url>,replication_endpoint=<endpoint>]
  --instance
fqdn=<cluster_fqdn>,username=<username>,password=<password>[,url=<url>,replication_endpoint=<endpoint>]
  [--port <port_number>]
  [--no-wait]
  [--default-db-config <configuration>]
  [--default-db-config-file <filename>]
  [--compression <0-6>]
  [--causal-consistency { true | false } ]
  [--password <password>]
  [--replication { true | false } ]
  [--encryption { true | false } ]
  [--sharding { false | true } ]
  [--shards-count <number_of_shards>]
  [--shard-key-regex <regex_rule>]
  [--oss-cluster { true | false } ]
  [--bigstore { true | false } ]
  [--bigstore-ram-size <maximum_memory>]
  [--with-module name=<module_name>,version=<module_version>,args=<module_args>]

```

Prerequisites

Before you create an Active-Active database, you must have:

- At least two participating clusters
- [Network connectivity](#) between the participating clusters

Parameters

Parameter & options(s)	Value	Description
name <CRDB_name>	string	Name of the Active-Active database (required)
memory-size <maximum_memory>	size in bytes, kilobytes (KB), or gigabytes (GB)	Maximum database memory (required)
instance		
fqdn=<cluster_fqdn>, username=<username>, password=<password>	strings	The connection information for the participating clusters (required for each participating cluster)
port <port_number>	integer	TCP port for the Active-Active database on all participating clusters
default-db-config <configuration>	string	Default database configuration options
default-db-config-file <filename>	filepath	Default database configuration options from a file
no-wait		Prevents <code>crdb-cli</code> from running another command before this command finishes
compression	0-6	The level of data compression: 0 = No compression 6 = High compression and resource load (Default: 3)
causal-consistency	true false (<i>default</i>)	Causal consistency applies updates to all instances in the order they were received
password <password>	string	Password for access to the database
replication	true false (<i>default</i>)	Activates or deactivates database replication where every master shard replicates to a replica shard
encryption	true false (<i>default</i>)	Activates or deactivates encryption
sharding	true false (<i>default</i>)	Activates or deactivates sharding (also known as database clustering). Cannot be updated after the database is created

Parameter & options(s)	Value	Description
shards-count <number_of_shards>	integer	If sharding is enabled, this specifies the number of Redis shards for each database instance
oss-cluster	true false (<i>default</i>)	Activates OSS cluster API
shard-key-regex <regex_rule>	string	If clustering is enabled, this defines a regex rule (also known as a hashing policy) that determines which keys are located in each shard (defaults to {u' regex': u'.*\\"{(?<tag>.*)\\"}.*'}, {u' regex': u'(?<tag>.*)' })
bigstore	true false (<i>default</i>)	If true, the database uses Auto Tiering to add flash memory to the database
bigstore-ram-size <size>	size in bytes, kilobytes (KB), or gigabytes (GB)	Maximum RAM limit for databases with Auto Tiering enabled
with-module name=<module_name>, version=<module_version>, args=<module_args>	strings	Creates a database with a specific module
eviction-policy	noeviction (<i>default</i>) allkeys-lru allkeys-lfu allkeys-random volatile-lru volatile-lfu volatile-random volatile-ttl all-nodes	Sets eviction policy
proxy-policy	all-master-shards single	Sets proxy policy

Returns

Returns the task ID of the task that is creating the database.

If `--no-wait` is specified, the command exits. Otherwise, it will wait for the database to be created and then return the CRDB GUID.

Examples

```
$ crdb-cli crdb create --name database1 --memory-size 1GB --port 12000 \
    --instance fqdn=cluster1.redis.local,username=admin@redis.local,password=admin \
    --instance fqdn=cluster2.redis.local,username=admin@redis.local,password=admin
Task 633aaea3-97ee-4bcb-af39-a9cb25d7d4da created
----> Status changed: queued -> started
----> CRDB GUID Assigned: crdb:d84f6fe4-5bb7-49d2-a188-8900e09c6f66
----> Status changed: started -> finished
```

To create an Active-Active database with two shards in each instance and with encrypted traffic between the clusters:

```
crdb-cli crdb create --name mycrdb --memory-size 100mb --port 12000 --instance
fqdn=cluster1.redis.local,username=admin@redis.local,password=admin --instance
fqdn=cluster2.redis.local,username=admin@redis.local,password=admin --shards-count 2 --encryption true
```

To create an Active-Active database with two shards and with RediSearch 2.0.6 module:

```
crdb-cli crdb create --name mycrdb --memory-size 100mb --port 12000 --instance
fqdn=cluster1.redis.local,username=admin@redis.local,password=admin --instance
fqdn=cluster2.redis.local,username=admin@redis.local,password=admin --shards-count 2 --with-module
name=search,version="2.0.6",args="PARTITIONS AUTO"
```

To create an Active-Active database with two shards and with encrypted traffic between the clusters:

```
crdb-cli crdb create --name mycrdb --memory-size 100mb --port 12000 --instance fqdn=cluster1.redis.local,username=admin@redis.local,password=admin --instance fqdn=cluster2.redis.local,username=admin@redis.local,password=admin --encryption true --shards-count 2
```

To create an Active-Active database with 1 shard in each instance and not wait for the response:

```
crdb-cli crdb create --name mycrdb --memory-size 100mb --port 12000 --instance fqdn=cluster1.redis.local,username=admin@redis.local,password=admin --instance fqdn=cluster2.redis.local,username=admin@redis.local,password=admin --no-wait
```

Updated: August 17, 2023

crdb-cli crdb delete

Deletes an Active-Active database.

```
crdb-cli crdb delete --crdb-guid <guid>
[ --no-wait ]
```

This command is irreversible. If the data in your database is important, back it up before you delete the database.

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)
no-wait		Does not wait for the task to complete

Returns

Returns the task ID of the task that is deleting the database.

If --no-wait is specified, the command exits. Otherwise, it will wait for the database to be deleted and return finished.

Example

```
$ crdb-cli crdb delete --crdb-guid db6365b5-8aca-4055-95d8-7eb0105c0b35
Task dfe6cacc-88ff-4667-812e-938fd05fe359 created
---> Status changed: queued -> started
---> Status changed: started -> finished
```

Updated: July 7, 2022

crdb-cli crdb flush

Clears all keys from an Active-Active database.

```
crdb-cli crdb flush --crdb-guid <guid>
[ --no-wait ]
```

This command is irreversible. If the data in your database is important, back it up before you flush the database.

Parameters

Parameter	Value	Description
-----------	-------	-------------

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)
no-wait		Does not wait for the task to complete

Returns

Returns the task ID of the task clearing the database.

If --no-wait is specified, the command exits. Otherwise, it will wait for the database to be cleared and return finished.

Example

```
$ crdb-cli crdb flush --crdb-guid d84f6fe4-5bb7-49d2-a188-8900e09c6f66
Task 53cdc59e-ecf5-4564-a8dd-448d71f9e568 created
---> Status changed: queued -> started
---> Status changed: started -> finished
```

Updated: July 7, 2022

crdb-cli crdb get

Shows the current configuration of an Active-Active database.

```
crdb-cli crdb get --crdb-guid <guid>
```

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)

Returns

Returns the current configuration of the database.

Example

```
$ crdb-cli crdb get --crdb-guid d84f6fe4-5bb7-49d2-a188-8900e09c6f66
CRDB-GUID: d84f6fe4-5bb7-49d2-a188-8900e09c6f66
Name: database1
Encryption: False
Causal consistency: False
Protocol version: 1
FeatureSet version: 5
Modules: []
Default-DB-Config:
  memory_size: 1073741824
  port: 12000
  replication: True
  shard_key_regex: [{"regex": ".*\\"\\{(?<tag>.*\\})\\\".*"}, {"regex": '(?<tag>.*)' }]
  sharding: True
  shards_count: 1
  tls_mode: disabled
  rack_aware: None
  data_persistence: None
  authentication_redis_pass: None
  authentication_admin_pass: None
  oss_sharding: None
  oss_cluster: None
  proxy_policy: None
  shards_placement: None
```

```
oss_cluster_api_preferred_ip_type: None
bigstore: None
bigstore_ram_size: None
aof_policy: None
snapshot_policy: None
max_aof_load_time: None
max_aof_file_size: None
Instance:
Id: 1
Cluster:
  FQDN: cluster1.redis.local
  URL: https://cluster1.redis.local:9443
  Replication-Endpoint: <Default>
  Replication TLS SNI: <Default>
Compression: 3
DB-Config:
  authentication_admin_pass:
  replication: None
  rack_aware: None
  memory_size: None
  data_persistence: None
  tls_mode: None
  authentication_redis_pass: None
  port: None
  shards_count: None
  shard_key_regex: None
  oss_sharding: None
  oss_cluster: None
  proxy_policy: None
  shards_placement: None
  oss_cluster_api_preferred_ip_type: None
  bigstore: None
  bigstore_ram_size: None
  aof_policy: None
  snapshot_policy: None
  max_aof_load_time: None
  max_aof_file_size: None
Instance:
Id: 2
Cluster:
  FQDN: cluster2.redis.local
  URL: https://cluster2.redis.local:9443
  Replication-Endpoint: <Default>
  Replication TLS SNI: <Default>
Compression: 3
DB-Config:
  authentication_admin_pass:
  replication: None
  rack_aware: None
  memory_size: None
  data_persistence: None
  tls_mode: None
  authentication_redis_pass: None
  port: None
  shards_count: None
  shard_key_regex: None
  oss_sharding: None
  oss_cluster: None
  proxy_policy: None
  shards_placement: None
  oss_cluster_api_preferred_ip_type: None
  bigstore: None
  bigstore_ram_size: None
  aof_policy: None
  snapshot_policy: None
  max_aof_load_time: None
  max_aof_file_size: None
```

crdb-cli crdb health-report

Shows the health report of the API management layer of an Active-Active database.

```
crdb-cli crdb health-report --crdb-guid <guid>
```

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)

Returns

Returns the health report of the API management layer of the database.

Example

```
$ crdb-cli crdb health-report --crdb-guid d84f6fe4-5bb7-49d2-a188-8900e09c6f66
[
  {
    "active_config_version":1,
    "cluster_name":"cluster2.redis.local",
    "configurations":[
      {
        "causal_consistency":false,
        "encryption":false,
        "featureset_version":5,
        "instances":[
          {
            "cluster":{
              "name":"cluster1.redis.local",
              "url":"https://cluster1.redis.local:9443"
            },
            "db_uid":"",
            "id":1
          },
          {
            "cluster":{
              "name":"cluster2.redis.local",
              "url":"https://cluster2.redis.local:9443"
            },
            "db_uid":"1",
            "id":2
          }
        ],
        "name":"database1",
        "protocol_version":1,
        "status":"commit-completed",
        "version":1
      }
    ],
    "connections":[
      {
        "name":"cluster1.redis.local",
        "status":"ok"
      },
      {
        "name":"cluster2.redis.local",
        "status":"ok"
      }
    ]
  }
]
```

```

],
"guid":"d84f6fe4-5bb7-49d2-a188-8900e09c6f66",
"name":"database1",
"connection_error":null
},
{
"active_config_version":1,
"cluster_name":"cluster1.redis.local",
"configurations":[
{
"causal_consistency":false,
"encryption":false,
"featureset_version":5,
"instances":[
{
"cluster":{
"name":"cluster1.redis.local",
"url":"https://cluster1.redis.local:9443"
},
"db_uid":"4",
"id":1
},
{
"cluster":{
"name":"cluster2.redis.local",
"url":"https://cluster2.redis.local:9443"
},
"db_uid":"",
"id":2
}
],
"name":"database1",
"protocol_version":1,
"status":"commit-completed",
"version":1
}
],
"connections":[
{
"name":"cluster1.redis.local",
"status":"ok"
},
{
"name":"cluster2.redis.local",
"status":"ok"
}
],
"guid":"d84f6fe4-5bb7-49d2-a188-8900e09c6f66",
"name":"database1",
"connection_error":null
}
]

```

Updated: November 22, 2022

crdb-cli crdb list

Shows a list of all Active-Active databases.

```
crdb-cli crdb list
```

Parameters

None

Returns

Returns a list of all Active-Active databases that the cluster participates in. Each database is represented with a unique GUID, the name of the database, an instance ID, and the FQDN of the cluster that hosts the instance.

Example

```
$ crdb-cli crdb list
CRDB-GUID          NAME      REPL-ID CLUSTER-FQDN
d84f6fe4-5bb7-49d2-a188-8900e09c6f66 database1 1      cluster1.redis.local
d84f6fe4-5bb7-49d2-a188-8900e09c6f66 database1 2      cluster2.redis.local
```

Updated: July 7, 2022

crdb-cli crdb purge-instance

Deletes data from a local instance and removes the instance from the Active-Active database.

```
crdb-cli crdb purge-instance --crdb-guid <guid>
    --instance-id <instance-id>
    [ --no-wait ]
```

Once this command finishes, the other replicas must remove this instance with [crdb-cli crdb remove-instance --force](#).

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)
instance-id	string	The ID of the local instance (required)
no-wait		Does not wait for the task to complete

Returns

Returns the task ID of the task that is purging the local instance.

If `--no-wait` is specified, the command exits. Otherwise, it will wait for the instance to be purged and return finished.

Example

```
$ crdb-cli crdb purge-instance --crdb-guid db6365b5-8aca-4055-95d8-7eb0105c0b35 --instance-id 2
Task add0705c-87f1-4c28-ad6a-ab5d98e00c58 created
----> Status changed: queued -> started
----> Status changed: started -> finished
```

Updated: July 7, 2022

crdb-cli crdb remove-instance

Removes a peer replica instance from the Active-Active database and deletes the instance and its data from the participating cluster.

```
crdb-cli crdb remove-instance --crdb-guid <guid>
    --instance-id <instance-id>
    [ --force ]
    [ --no-wait ]
```

If the cluster cannot communicate with the instance that you want to remove, you can:

1. Use the `--force` option to remove the instance from the Active-Active database without purging the data from the instance.
2. Run `crdb-cli crdb purge-instance` from the removed instance to delete the Active-Active database and its data.

Parameters

Parameter	Value	Description
crdb-guid	string	The GUID of the database (required)
instance-id	string	The ID of the local instance to remove (required)
force		Removes the instance without purging data from the instance. If <code>--force</code> is specified, you must run <code>crdb-cli crdb purge-instance</code> from the removed instance.
no-wait		Does not wait for the task to complete

Returns

Returns the task ID of the task that is deleting the instance.

If `--no-wait` is specified, the command exits. Otherwise, it will wait for the instance to be removed and return `finished`.

Example

```
$ crdb-cli crdb remove-instance --crdb-guid db6365b5-8aca-4055-95d8-7eb0105c0b35 --instance-id 2 --force
Task b1eba5ba-90de-49e9-8678-d66daa1afb51 created
    ---> Status changed: queued -> started
    ---> Status changed: started -> finished
```

Updated: July 7, 2022

crdb-cli crdb update

Updates the configuration of an Active-Active database.

```
crdb-cli crdb update --crdb-guid <guid>
  [--no-wait]
  [--force]
  [--default-db-config <configuration> ]
  [--default-db-config-file <filename>]
  [--compression <0-6>]
  [--causal-consistency { true | false } ]
  [--credentials id=<id>,username=<username>,password=<password> ]
  [--encryption { true | false } ]
  [--oss-cluster { true | false } ]
  [--featureset-version { true | false } ]
  [--memory-size <maximum_memory>]
  [--bigstore-ram-size <maximum_memory>]
  [--update-module name=<name>,featureset_version=<version>]
```

If you want to change the configuration of the local instance only, use `r1admin` instead.

Parameters

Parameter	Value	Description
crdb-guid <guid>	string	GUID of the Active-Active database (required)

Parameter	Value	Description
bigstore-ram-size <maximum_memory>	size in bytes, kilobytes (KB), or gigabytes (GB)	Maximum RAM limit for the databases with Auto Tiering enabled, if activated
memory-size <maximum_memory>	size in bytes, kilobytes (KB), or gigabytes (GB)	Maximum database memory (required)
causal-consistency	true false	Causal consistency applies updates to all instances in the order they were received
		The level of data compression:
compression	0-6	0 = No compression 6 = High compression and resource load (Default: 3)
credentials id=<id>,username=<username>,password=<password>	strings	Updates the credentials for access to the instance
default-db-config <configuration>		Default database configuration from stdin
default-db-config-file <filename>	filepath	Default database configuration from file
encryption	true false	Activates or deactivates encryption
force		Force an update even if there are no changes
no-wait		Do not wait for the command to finish
oss-cluster	true false	Activates or deactivates OSS Cluster mode
eviction-policy	noeviction allkeys-lru allkeys-lfu allkeys-random volatile-lru volatile-lfu volatile-random volatile-ttl	Updates eviction policy
featureset-version	true false	Updates to latest FeatureSet version
update-module name=<name>,featureset_version=<version>	strings	Update a module to the specified version

Returns

Returns the task ID of the task that is updating the database.

If `--no-wait` is specified, the command exits. Otherwise, it will wait for the database to be updated and then return "finished."

Example

```
$ crdb-cli crdb update --crdb-guid 968d586c-e12d-4b8f-8473-42eb88d0a3a2 --memory-size 2GBTask 7e98efc1-8233-4578-9e0c-cdc854b8af9e created
---> Status changed: queued -> started
---> Status changed: started -> finished
```

Updated: August 17, 2023

crdb-cli task commands

The `crdb-cli task` commands help investigate Active-Active database performance issues. They should not be used except as directed by Support.

crdb-cli task commands

Command	Description
---------	-------------

Command	Description
cancel	Attempts to cancel a specified Active-Active database task.
list	Lists active and recent Active-Active database tasks.
status	Shows the status of a specified Active-Active database task.

Updated: October 13, 2022

crdb-cli task cancel

Cancels the Active-Active database task specified by the task ID.

```
crdb-cli task cancel <task_id>
```

Parameters

Parameter	Value	Description
task-id <task_id>	string	An Active-Active database task ID (required)

Returns

Attempts to cancel an Active-Active database task.

Be aware that tasks may complete before they can be cancelled.

Example

```
$ crdb-cli task cancel --task-id 2901c2a3-2828-4717-80c0-6f27f1dd2d7c
```

Updated: October 13, 2022

crdb-cli task list

Lists active and recent Active-Active database tasks.

```
crdb-cli task list
```

Parameters

None

Returns

A table listing current and recent Active-Active tasks. Each entry includes the following:

Column	Description
Task ID	String containing the unique ID associated with the task Example: e1c49470-ae0b-4df8-885b-9c755dd614d0
CRDB-GUID	String containing the unique ID associated with the Active-Active database affected by the task Example: 1d7741cc-1110-4e2f-bc6c-935292783d24
Operation	String describing the task action Example: create_crd

Column	Description
Status	String indicating the task status Example: finished
Worker name	String identifying the process handling the task Example: crdb_worker:1:0
Started	TimeStamp value indicating when the task started (UTC) Example: 2022-10-12T09:33:41Z
Ended	TimeStamp value indicating when the task ended (UTC) Example: 2022-10-12T09:33:55Z

Example

```
$ crdb-cli task list
TASK-ID    CRDB-GUID  OPERATION    STATUS     WORKER-NAME  STARTED    ENDED
<task-ID>  <crdb-ID>  <operation>  <result>   <worker-ID> <started>   <ended>
```

Updated: October 13, 2022

crdb-cli task status

Shows the status of a specified Active-Active database task.

```
crdb-cli task status --task-id <task_id>
```

Parameters

Parameter	Value	Description
task-id <task_id>	string	An Active-Active database task ID (required)
verbose	N/A	Returns detailed information when specified
no-verbose	N/A	Returns limited information when specified

The `--verbose` and `--no-verbose` options are mutually incompatible; specify one or the other.

The `404 Not Found` error indicates an invalid task ID. Use the [task list](#) command to determine available task IDs.

Returns

Returns the status of an Active-Active database task.

Example

```
$ crdb-cli task status --task-id e1c49470-ae0b-4df8-885b-9c755dd614d0
Task-ID: e1c49470-ae0b-4df8-885b-9c755dd614d0
CRDB-GUID: 1d7741cc-1110-4e2f-bc6c-935292783d24
Operation: create_crdb
Status: finished
Worker-Name: crdb_worker:1:0
Started: 2022-10-12T09:33:41Z
Ended: 2022-10-12T09:33:55Z
```

Updated: October 13, 2022

redis-cli

The `redis-cli` command-line utility lets you interact with a Redis database. With `redis-cli`, you can run [Redis commands](#) directly from the command-line terminal or with [interactive mode](#).

If you want to run Redis commands without `redis-cli`, you can [connect to a database with RedisInsight](#) and use the built-in [CLI](#) prompt instead.

Install `redis-cli`

When you install Redis Enterprise Software or open source Redis, it also installs the `redis-cli` command-line utility.

To learn how to install Redis and `redis-cli`, see the following installation guides:

- [Open source Redis](#)
- [Redis Enterprise Software](#)
- [Redis Enterprise Software with Docker](#)

Connect to a database

To run Redis commands with `redis-cli`, you need to connect to your Redis database.

You can find endpoint and port details in the **Databases** list or the database's **Configuration** screen.

Connect remotely

If you have `redis-cli` installed on your local machine, you can use it to connect to a remote Redis database. You will need to provide the database's connection details, such as the hostname or IP address, port, and password.

```
$ redis-cli -h <endpoint> -p <port> -a <password>
```

You can also provide the password with the `REDISCLI_AUTH` environment variable instead of the `-a` option:

```
$ export REDISCLI_AUTH=<password>
$ redis-cli -h <endpoint> -p <port>
```

Connect over TLS

To connect to a Redis Enterprise Software or Redis Enterprise Cloud database over TLS:

1. Download or copy the Redis Enterprise server (or proxy) certificates.
 - For Redis Enterprise Cloud, see [Download certificates](#) for detailed instructions on how to download the server certificates (`redis_ca.pem`) from the [admin console](#).
 - For Redis Enterprise Software, copy the proxy certificate from the admin console (**Cluster > Security > Certificates > Server authentication**) or from a cluster node (`/etc/opt/redislabs/proxy_cert.pem`).
2. If your database doesn't require client authentication, provide the Redis Enterprise server certificate (`redis_ca.pem` for Cloud or `proxy_cert.pem` for Software) when you connect:

```
redis-cli -h <endpoint> -p <port> --tls --cacert <redis_cert>.pem
```

3. If your database requires client authentication, then you also need to provide your client's private and public keys:

```
redis-cli -h <endpoint> -p <port> --tls --cacert <redis_cert>.pem \
--cert redis_user.crt --key redis_user_private.key
```

Connect with Docker

If your Redis database runs in a Docker container, you can use `docker exec` to run `redis-cli` commands:

```
$ docker exec -it <Redis container name> redis-cli -p <port>
```

Basic use

You can run `redis-cli` commands directly from the command-line terminal:

```
$ redis-cli -h <endpoint> -p <port> <Redis command>
```

For example, you can use `redis-cli` to test your database connection and store a new Redis string in the database:

```
$ redis-cli -h <endpoint> -p 12000 PING
PONG
$ redis-cli -h <endpoint> -p 12000 SET mykey "Hello world"
OK
$ redis-cli -h <endpoint> -p 12000 GET mykey
"Hello world"
```

For more information, see [Command line usage](#).

Interactive mode

In `redis-cli` [interactive mode](#), you can:

- Run any `redis-cli` command without prefacing it with `redis-cli`.
- Enter `?` for more information about how to use the `HELP` command and [set `redis-cli` preferences](#).
- Enter `HELP` followed by the name of a command for more information about the command and its options.
- Press the Tab key for command completion.
- Enter `exit` or `quit` or press Control+D to exit interactive mode and return to the terminal prompt.

This example shows how to start interactive mode and run Redis commands:

```
$ redis-cli -p 12000
127.0.0.1:12000> PING
PONG
127.0.0.1:12000> SET mykey "Hello world"
OK
127.0.0.1:12000> GET mykey
"Hello world"
```

Examples

Check slowlog

Run `slowlog get` for a list of recent slow commands:

```
redis-cli -h <endpoint> -p <port> slowlog get <number of entries>
```

Scan for big keys

Scan the database for big keys:

```
redis-cli -h <endpoint> -p <port> --bigkeys
```

See [Scanning for big keys](#) for more information.

More info

- [Redis CLI documentation](#)
- [Redis commands reference](#)

Updated: August 17, 2023

`rladmin` is a command-line utility that lets you perform administrative tasks such as failover, migration, and endpoint binding on a Redis Enterprise Software cluster. You can also use `rladmin` to edit cluster and database configurations.

Although you can use the admin console for some of these tasks, others are unique to the `rladmin` command-line tool.

rladmin commands

Command	Description
bind	Manages the proxy policy for a specified database endpoint.
cluster	Manage cluster.
failover	Fail over primary shards of a database to their replicas.
help	Shows available commands or specific command usage.
info	Shows the current configuration of a cluster, database, node, or proxy.
migrate	Moves Redis Enterprise Software shards or endpoints to a new node in the same cluster.
node	Manage nodes.
placement	Configures the shard placement policy for a database.
recover	Recovers databases in recovery mode.
restart	Restarts Redis Enterprise Software processes for a specific database.
status	Displays the current cluster status and topology information.
suffix	Manages the DNS suffixes in the cluster.
tune	Configures parameters for databases, proxies, nodes, and clusters.
upgrade	Upgrades the version of a module or Redis Enterprise Software for a database.
verify	Prints verification reports for the cluster.

Use the `rladmin` shell

To open the `rladmin` shell:

1. Sign in to a Redis Enterprise Software node with an account that is a member of the `redislabs` group.

The `rladmin` binary is located in `/opt/redislabs/bin`. If you don't have this directory in your PATH, you may want to add it. Otherwise, you can use `bash -l <username>` to sign in as a user with permissions for that directory.

2. Run: `rladmin`



Note: If the CLI does not recognize the `rladmin` command, run this command to load the necessary configuration first: `bash -l`

In the `rladmin` shell, you can:

- Run any `rladmin` command without prefacing it with `rladmin`.
- Enter `?` to view the full list of available commands.
- Enter `help` followed by the name of a command for a detailed explanation of the command and its usage.
- Press the Tab key for command completion.
- Enter `exit` or press Control+D to exit the `rladmin` shell and return to the terminal prompt.

Updated: July 7, 2022

rladmin bind

Manages the proxy policy for a specific database endpoint.

bind endpoint exclude

Defines a list of nodes to exclude from the proxy policy for a specific database endpoint. When you exclude a node, the endpoint cannot bind to the node's proxy.

Each time you run an exclude command, it overwrites the previous list of excluded nodes.

```
rladmin bind
  [ db { db:<id> | <name> } ]
  endpoint <id> exclude
  <proxy_id1 .. proxy_idN>
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Only allows endpoints for the specified database
endpoint	endpoint ID	Changes proxy settings for the specified endpoint
proxy	list of proxy IDs	Proxies to exclude

Returns

Returns Finished successfully if the list of excluded proxies was successfully changed. Otherwise, it returns an error.

Use [rladmin status endpoints](#) to verify that the policy changed.

Example

```
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID  NAME   ID           NODE      ROLE          SSL
db:6    tr02   endpoint:6:1  node:2    all-nodes    No
db:6    tr02   endpoint:6:1  node:1    all-nodes    No
db:6    tr02   endpoint:6:1  node:3    all-nodes    No
$ rladm bind endpoint 6:1 exclude 2
Executing bind endpoint: 000.
Finished successfully
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID  NAME   ID           NODE      ROLE          SSL
db:6    tr02   endpoint:6:1  node:1    all-nodes -2  No
db:6    tr02   endpoint:6:1  node:3    all-nodes -2  No
```

bind endpoint include

Defines a list of nodes to include in the proxy policy for the specific database endpoint.

Each time you run an include command, it overwrites the previous list of included nodes.

```
rladmin bind
  [ db { db:<id> | <name> } ]
  endpoint <id> include
  <proxy_id1 .. proxy_idN>
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Only allows endpoints for the specified database
endpoint	endpoint ID	Changes proxy settings for the specified endpoint
proxy	list of proxy IDs	Proxies to include

Returns

Returns Finished successfully if the list of included proxies was successfully changed. Otherwise, it returns an error.

Use `rladmin status endpoints` to verify that the policy changed.

Example

```
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID  NAME   ID           NODE    ROLE          SSL
db:6    tr02   endpoint:6:1  node:3  all-master-shards No
$ rladm bind endpoint 6:1 include 3
Executing bind endpoint: 000.
Finished successfully
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID  NAME   ID           NODE    ROLE          SSL
db:6    tr02   endpoint:6:1  node:1  all-master-shards +3 No
db:6    tr02   endpoint:6:1  node:3  all-master-shards +3 No
```

bind endpoint policy

Changes the overall proxy policy for a specific database endpoint.

```
rladmin bind
  [ db { db:<id> | <name> } ]
  endpoint <id>
  policy { single | all-master-shards | all-nodes }
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Only allows endpoints for the specified database
endpoint	endpoint ID	Changes proxy settings for the specified endpoint
policy	'all-master-shards' 'all-nodes' 'single'	Changes the proxy policy to the specified policy

Proxy policy	Description
all-master-shards	Multiple proxies, one on each master node (best for high traffic and multiple master shards)
all-nodes	Multiple proxies, one on each node of the cluster (increases traffic in the cluster, only used in special cases)
single	All traffic flows through a single proxy bound to the database endpoint (preferable in most cases)

Returns

Returns `Finished successfully` if the proxy policy was successfully changed. Otherwise, it returns an error.

Use `rladmin status endpoints` to verify that the policy changed.

Example

```

$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID    NAME    ID          NODE      ROLE           SSL
db:6     tr02   endpoint:6:1  node:1    all-nodes -2  No
db:6     tr02   endpoint:6:1  node:3    all-nodes -2  No
$ rladm bind endpoint 6:1 policy all-master-shards
Executing bind endpoint: 000.
Finished successfully
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID    NAME    ID          NODE      ROLE           SSL
db:6     tr02   endpoint:6:1  node:3    all-master-shards No

```

Updated: June 10, 2022

rladmin cluster

Manages cluster configuration and administration. Most `rladmin cluster` commands are only for clusters that are already configured, while a few others are only for new clusters that have not been configured.

Commands for configured clusters

Command	Description
certificate	Sets the cluster certificate.
config	Updates the cluster's configuration.
debug_info	Creates a support package.
master	Identifies or changes the cluster's master node.
ocsp	Manages OCSP.
reset_password	Changes the password for a given email.
running_actions	Lists all active tasks.
stats_archiver	Enables/deactivates the stats archiver.

Commands for non-configured clusters

Command	Description
create	Creates a new cluster.
join	Adds a node to an existing cluster.
recover	Recovers a cluster from a backup file.

Updated: June 10, 2022

rladmin cluster certificate

Sets a cluster certificate to a specified PEM file.

```

rladmin cluster certificate
  set <certificate name>
  certificate_file <certificate filepath>
  [ key_file <key filepath> ]

```

To set a certificate for a specific service, use the corresponding certificate name. See the [certificates table](#) for the list of cluster certificates and their

descriptions.

Parameters

Parameter	Type/Value	Description
certificate name	'cm' 'api' 'proxy' 'syncer' 'metrics_exporter'	Name of the certificate to update
certificate_file	filepath	Path to the certificate file
key_file	filepath	Path to the key file (optional)

Returns

Reports that the certificate was set to the specified file. Returns an error message if the certificate fails to update.

Example

```
$ rladm cluster certificate set proxy \
  certificate_file /tmp/proxy.pem
Set proxy certificate to contents of file /tmp/proxy.pem
```

Updated: December 23, 2022

rladmin cluster config

Updates the cluster configuration.

```
rladmin cluster config
  [ auditing db_conns audit_protocol { TCP | local }
    audit_address <audit_address> audit_port <audit_port> ]
  [ bigstore_driver {speedb | rocksdb} ]
  [ cipher_suites <BoringSSL cipher list> ]
  [ cm_port <number> ]
  [ cm_session_timeout <minutes> ]
  [ cnm_http_port <number> ]
  [ cnm_https_port <number> ]
  [ data_cipher_list <openSSL cipher list> ]
  [ debuginfo_path <filepath> ]
  [ encrypt_pkeys { enabled | disabled } ]
  [ envoy_admin_port <new-port> ]
  [ envoy_mgmt_server_port <new-port> ]
  [ gossip_envoy_admin_port <new-port> ]
  [ handle_redirects { enabled | disabled } ]
  [ handle_metrics_redirects { enabled | disabled } ]
  [ http_support { enabled | disabled } ]
  [ ipv6 { enabled | disabled } ]
  [ min_control_TLS_version <control_tls_version> ]
  [ min_data_TLS_version <data_tls_version> ]
  [ min_sentinel_TLS_version <sentinel_tls_version> ]
  [ s3_url <URL> ]
  [ saslauthd_ldap_conf </tmp/ldap.conf> ]
  [ sentinel_ssl_policy { allowed | required | disabled } ]
  [ sentinel_cipher_suites <golang cipher list> ]
  [ services { cm_server | crdb_coordinator | crdb_worker |
    mdns_server | pdns_server | saslauthd |
    stats_archiver } { enabled | disabled } ]
  [ upgrade_mode { enabled | disabled } ]
```

Parameters

Parameter	Type/Value	Description
audit_address	string	TCP/IP address where a listener can capture audit event notifications
audit_port	string	Port where a listener can capture audit event notifications
audit_protocol	tcp local	Protocol used for audit event notifications For production systems, only tcp is supported.
cipher_suites	list of ciphers	Cipher suites used for TLS connections to the admin console (specified in the format understood by the BoringSSL library)
cm_port	integer	UI server listening port
cm_session_timeout	integer	Timeout in minutes for the CM session
cmn_http_port	integer	HTTP REST API server listening port
cnm_https_port	integer	HTTPS REST API server listening port
data_cipher_list	list of ciphers	Cipher suites used by the the data plane (specified in the format understood by the OpenSSL library)
debuginfo_path	filepath	Local directory to place generated support package files
encrypt_pkeys	enabled disabled	Enable or turn off encryption of private keys
envoy_admin_port	integer, (range: 1024-65535)	Envoy admin port. Changing this port during runtime might result in an empty response because envoy serves as the cluster gateway.
envoy_mgmt_server_port	integer, (range: 1024-65535)	Envoy management server port
gossip_envoy_admin_port	integer, (range: 1024-65535)	Gossip envoy admin port
handle_redirects	enabled disabled	Enable or turn off handling DNS redirects when DNS is not configured and running behind a load balancer
handle_metrics_redirects	enabled disabled	Enable or turn off handling cluster redirects internally for Metrics API
http_support	enabled disabled	Enable or turn off using HTTP for REST API connections
ipv6	enabled disabled	Enable or turn off IPv6 connections to the admin console
min_control_TLS_version	TLS protocol version	The minimum TLS protocol version that is supported for the control path
min_data_TLS_version	TLS protocol version	The minimum TLS protocol version that is supported for the data path
min_sentinel_TLS_version	TLS protocol version	The minimum TLS protocol version that is supported for the discovery service
s3_url	string	The URL of S3 export and import
saslauthd_ldap_conf	filepath	Updates LDAP authentication configuration for the cluster
sentinel_cipher_suites	list of ciphers	Cipher suites used by the discovery service (supported ciphers are implemented by the golang.org cipher suites package)
sentinel_ssl_policy	allowed required disabled	Define the SSL policy for the discovery service

Parameter	Type/Value	Description
services	cm_server crdb_coordinator crdb_worker mdns_server pdns_server saslauthd stats_archiver	Enable or turn off selected cluster services
upgrade_mode	enabled disabled	enabled disabled

Returns

Reports whether the cluster was configured successfully. Displays an error message if the configuration attempt fails.

Example

```
$ rladm cluster config cm_session_timeout_minutes 20
Cluster configured successfully
```

Updated: August 17, 2023

rladmin cluster create

Creates a new cluster. The node where you run `rladmin cluster create` becomes the first node of the new cluster.

```
cluster create
  name <cluster name>
  username <admin email>
  password <admin password>
  [ node_uid <node UID> ]
  [ rack_aware ]
  [ rack_id <node rack ID> ]
  [ license_file <file> ]
  [ ephemeral_path <path> ]
  [ persistent_path <path> ]
  [ ccs_persistent_path <path> ]
  [ register_dns_suffix ]
  [ flash_enabled ]
  [ flash_path <path> ]
  [ addr <IP address> ]
  [ external_addr <IP address> ]
```

Parameters

Parameter	Type/Value	Description
addr	IP address	The node's internal IP address (optional)
ccs_persistent_path	filepath (default: /var/opt/redislabs/persist)	Path to the location of CCS snapshots (optional)
ephemeral_path	filepath (default: /var/opt/redislabs)	Path to the ephemeral storage location (optional)
external_addr	IP address	The node's external IP address (optional)
flash_enabled		Enables flash storage (optional)
flash_path	filepath (default: /var/opt/redislabs/flash)	Path to the flash storage location (optional)
license_file	filepath	Path to the RLEC license file (optional)
name	string	Cluster name

Parameter	Type/Value	Description
node_uid	integer	Unique node ID (optional)
password	string	Admin user's password
persistent_path	filepath (default: /var/opt/redislabs/persist)	Path to the persistent storage location (optional)
rack_aware		Activates or deactivates rack awareness (optional)
rack_id	string	The rack's unique identifier (optional)
register_dns_suffix		Enables database mapping to both internal and external IP addresses (optional)
username	email address	Admin user's email address

Returns

Returns ok if the new cluster was created successfully. Otherwise, it returns an error message.

Example

```
$ rladm cluster create name cluster.local \
    username admin@example.com \
    password admin-password
Creating a new cluster... ok
```

Updated: July 7, 2022

rladmin cluster debug_info

Downloads a support package to the specified path. If you do not specify a path, it downloads the package to the default path specified in the cluster configuration file.

```
rladmin cluster debug_info
  [ node <ID> ]
  [ path <path> ]
  [ sanitized ]
```

Parameters

Parameter	Type/Value	Description
node	integer	Downloads a support package for the specified node
path	filepath	Specifies the location where the support package should download
sanitized		Removes sensitive data (passwords, certificates, etc.) from the support package

Returns

Reports the progress of the support package download.

Example

```
$ rladm cluster debug_info node 1 sanitized
Preparing the debug info files package
Downloading...
[=====]
Downloading complete. File /tmp/debuginfo.20220511-215637.node-1.tar.gz is saved.
```

rladmin cluster join

Adds a node to an existing cluster.

```
rladmin cluster join
    nodes <node IP address>
    username <admin user email>
    password <admin password>
    [ ephemeral_path <path> ]
    [ persistent_path <path> ]
    [ ccs_persistent_path <path> ]
    [ rack_id <node rack ID> ]
    [ override_rack_id ]
    [ replace_node <node UID> ]
    [ flash_enabled ]
    [ flash_path <path> ]
    [ addr <IP address> ]
    [ external_addr <IP address> ]
    [ override_repair ]
    [ accept_servers { enabled | disabled } ]
    [ cmn_http_port <port> ]
```

Parameters

Parameter	Type/Value	Description
accept_servers	'enabled' 'disabled'	Allows allocation of resources on the new node when enabled (optional)
addr	IP address	Sets a node's internal IP address. If not provided, the node sets the address automatically. (optional)
ccs_persistent_path	filepath (default: /var/opt/redislabs/persist)	Path to the CCS snapshot location (the default is the same as persistent_path) (optional)
cmn_http_port	integer	Joins a cluster that has a non-default cmn_http_port (optional)
ephemeral_path	filepath	Path to the ephemeral storage location (optional)
external_addr	IP address	Sets a node's external IP address. If not provided, the node sets the address automatically. (optional)
flash_enabled		Enables flash capabilities for a database (optional)
flash_path	filepath (default: /var/opt/redislabs/flash)	Path to the flash storage location in case the node does not support CAPI (required if flash_enabled)
nodes	IP address	Internal IP address of an existing node in the cluster
override_rack_id		Changes to a new rack, specified by rack_id (optional)
override_repair		Enables joining a cluster with a dead node (optional)
password	string	Admin user's password
persistent_path	filepath (default: /var/opt/redislabs/persist)	Path to the persistent storage location (optional)
rack_id	string	Moves the node to the specified rack (optional)
replace_node	integer	Replaces the specified node with the new node (optional)
username	email address	Admin user's email address

Returns

Returns ok if the node joined the cluster successfully. Otherwise, it returns an error message.

Example

```
$ rladm cluster join nodes 192.0.2.2 \
    username admin@example.com \
    password admin-password
Joining cluster... ok
```

Updated: July 7, 2022

rladmin cluster master

Identifies the cluster's master node. Use `set` to change the cluster's master to a different node.

```
cluster master [ set <node_id> ]
```

Parameters

Parameter	Type/Value	Description
<code>node_id</code>	integer	Unique node ID

Returns

Returns the ID of the cluster's master node. Otherwise, it returns an error message.

Example

Identify the cluster's master node:

```
$ rladm cluster master
Node 1 is the cluster master node
```

Change the cluster master to node 3:

```
$ rladm cluster master set 3
Node 3 set to be the cluster master node
```

Updated: August 24, 2023

rladmin cluster ocsp

Manages OCSP configuration and verifies the status of a server certificate maintained by a third-party [certificate authority \(CA\)](#).

ocsp certificate_compatible

Checks if the proxy certificate contains an OCSP URI.

```
rladmin cluster ocsp certificate_compatible
```

Parameters

None

Returns

Returns the OCSP URI if it exists. Otherwise, it returns an error.

Example

```
$ rladm cluster ocsp certificate_compatible  
Success. OCSP URI is http://responder.ocsp.url.com
```

ocsp config

Displays or updates OCSP configuration. Run the command without the set option to display the current configuration of a parameter.

```
rladmin cluster ocsp config <OCSP parameter>  
[ set <value> ]
```

Parameters

Parameter	Type/Value	Description
ocsp_functionality	enabled	Enables or turns off OCSP for the cluster
	disabled	
query_frequency	integer (range: 60-86400) (default: 3600)	The time interval in seconds between OCSP queries to check the certificate's status
recovery_frequency	integer (range: 60-86400) (default: 60)	The time interval in seconds between retries after a failed query
recovery_max_tries	integer (range: 1-100) (default: 5)	The number of retries before the validation query fails and invalidates the certificate
responder_url	string	The OCSP server URL embedded in the proxy certificate (you cannot manually set this parameter)
response_timeout	integer (range: 1-60) (default: 1)	The time interval in seconds to wait for a response before timing out

Returns

If you run the `ocsp config` command without the `set` option, it displays the specified parameter's current configuration.

Example

```
$ rladm cluster ocsp config recovery_frequency  
Recovery frequency of the OCSP server is 60 seconds  
$ rladm cluster ocsp config recovery_frequency set 30  
$ rladm cluster ocsp config recovery_frequency  
Recovery frequency of the OCSP server is 30 seconds
```

ocsp status

Returns the latest cached status of the certificate's OCSP response.

```
rladmin cluster ocsp status
```

Parameters

None

Returns

Returns the latest cached status of the certificate's OCSP response.

Example

```
$ rladm cluster ocsp status
OCSP certificate status is: REVOKED
produced_at: Wed, 22 Dec 2021 12:50:11 GMT
responder_url: http://responder.ocsp.url.com
revocation_time: Wed, 22 Dec 2021 12:50:04 GMT
this_update: Wed, 22 Dec 2021 12:50:11 GMT
```

ocsp test_certificate

Queries the OCSP server for the certificate's latest status, then caches and displays the response.

```
rladmin cluster ocsp test_certificate
```

Parameters

None

Returns

Returns the latest status of the certificate's OCSP response.

Example

```
$ rladm cluster ocsp test_certificate
Initiating a query to OCSP server
...OCSP certificate status is: REVOKED
produced_at: Wed, 22 Dec 2021 12:50:11 GMT
responder_url: http://responder.ocsp.url.com
revocation_time: Wed, 22 Dec 2021 12:50:04 GMT
this_update: Wed, 22 Dec 2021 12:50:11 GMT
```

Updated: August 10, 2022

rladmin cluster recover

Recover a cluster from a backup file. The default location of the configuration backup file is /var/opt/redislabs/persist/ccs/ccs-redis.rdb.

```
rladmin cluster recover
  filename <recovery filename>
  [ ephemeral_path <path> ]
  [ persistent_path <path> ]
  [ ccs_persistent_path <path> ]
  [ rack_id <ID> ]
  [ override_rack_id ]
  [ node_uid <number> ]
  [ flash_enabled ]
  [ flash_path <path> ]
  [ addr <IP address> ]
  [ external_addr <IP address> ]
```

Parameters

Parameter	Type/Value	Description
addr	IP address	Sets a node's internal IP address. If not provided, the node sets the address automatically. (optional)
ccs_persistent_path	filepath	Path to the location of CCS snapshots (default is the same as persistent_path) (optional)

Parameter	Type/Value	Description
external_addr	IP address	Sets a node's external IP address. If not provided, the node sets the address automatically. (optional)
ephemeral_path	filepath (default: /var/opt/redislabs)	Path to an ephemeral storage location (optional)
filename	filepath	Backup file to use for recovery
flash_enabled		Enables flash storage (optional)
flash_path	filepath (default: /var/opt/redislabs/flash)	Path to the flash storage location in case the node does not support CAPI (required if flash_enabled)
node_uid	integer (default: 1)	Specifies which node will recover first and become master (optional)
override_rack_id		Changes to a new rack, specified by rack_id (optional)
persistent_path	filepath	Path to the persistent storage location (optional)
rack_id	string	Switches to the specified rack (optional)

Returns

Returns ok if the cluster recovered successfully. Otherwise, it returns an error message.

Example

```
$ rladm cluster recover filename /tmp/persist/ccs/ccs-redis.rdb node_uid 1 rack_id 5
Initiating cluster recovery... ok
```

Updated: July 7, 2022

rladmin cluster reset_password

Changes the password for the user associated with the specified email address.

Enter a new password when prompted. Then enter the same password when prompted a second time to confirm the password change.

```
rladmin cluster reset_password <user email>
```

Parameters

Parameter	Type/Value	Description
user email	email address	The email address of the user that needs a password reset

Returns

Reports whether the password change succeeded or an error occurred.

Example

```
$ rladm cluster reset_password user@example.com
New password:
New password (again):
Password changed.
```

Updated: June 10, 2022

rladmin cluster running_actions

Lists all active tasks running on the cluster.

```
rladmin cluster running_actions
```

Parameters

None

Returns

Returns details about any active tasks running on the cluster.

Example

```
$ rladm cluster running_actions
Got 1 tasks:
1) Task: maintenance_on (ce391d81-8d51-4ce2-8f63-729c7ac2589e) Node: 1 Status: running
```

Updated: June 10, 2022

rladmin cluster stats_archiver

Enables or deactivates the stats archiver, which logs statistics in CSV (comma-separated values) format.

```
rladmin cluster stats_archiver { enabled | disabled }
```

Parameters

Parameter	Description
enabled	Turn on the stats archiver
disabled	Turn off the stats archiver

Returns

Returns the updated status of the stats archiver.

Example

```
$ rladm cluster stats_archiver enabled
Status: enabled
```

Updated: June 10, 2022

rladmin failover

Fails over one or more primary (also known as master) shards of a database and promotes their respective replicas to primary shards.

```
rladmin failover
  [db { db:<id> | <name> }]
  shard <id1 ... idN>
  [immediate]
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Fail over shards for the specified database
shard	one or more primary shard IDs	Primary shard or shards to fail over
immediate		Perform failover without verifying the replica shards are in full sync with the master shards

Returns

Returns Finished successfully if the failover completed. Otherwise, it returns an error.

Use [rladmin status shards](#) to verify that the failover completed.

Example

```
$ rladm status shards
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:5 tr01 redis:12 node:1 slave 0-16383 3.02MB OK
db:5 tr01 redis:13 node:2 master 0-16383 3.09MB OK
$ rladm failover shard 13
Executing shard fail-over: 000.
Finished successfully
$ rladm status shards
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:5 tr01 redis:12 node:1 master 0-16383 3.12MB OK
db:5 tr01 redis:13 node:2 slave 0-16383 2.99MB OK
```

Updated: June 10, 2022

rladmin help

Lists all options and parameters for `rladmin` commands.

```
rladmin help [command]
```

Parameters

Parameter	Description
command	Display help for this <code>rladmin</code> command (optional)

Returns

Returns a list of available `rladmin` commands.

If a command is specified, returns a list of all the options and parameters for that `rladmin` command.

Example

```
$ rladm help
usage: rladm [options] [command] [command args]

Options:
  -y Assume Yes for all required user confirmations.

Commands:
  bind      Bind an endpoint
  cluster   Cluster management commands
  exit      Exit admin shell
  failover  Fail-over master to slave
  help      Show available commands, or use help <command> for a specific command
  info      Show information about tunable parameters
  migrate   Migrate elements between nodes
  node      Node management commands
  placement Configure shards placement policy
  recover   Recover databases
  restart   Restart database shards
  status    Show status information
  suffix    Suffix management
  tune      Tune system parameters
  upgrade   Upgrade entity version
  verify    Cluster verification reports
```

Use "rladmin help [command]" to get more information on a specific command.

Updated: June 10, 2022

rladmin info

Shows the current configuration of specified databases, proxies, clusters, or nodes.

info cluster

Lists the current configuration for the cluster.

```
rladmin info cluster
```

Parameters

None

Returns

Returns the current configuration for the cluster.

Example

```
$ rladm info cluster
Cluster configuration:
  repl_diskless: enabled
  shards_overbooking: disabled
  default_non_sharded_proxy_policy: single
  default_sharded_proxy_policy: single
  default_shards_placement: dense
  default_fork_evict_ram: enabled
  default_provisioned_redis_version: 6.0
  redis_migrate_node_threshold: 0KB (0 bytes)
  redis_migrate_node_threshold_percent: 4 (%)
  redis_provision_node_threshold: 0KB (0 bytes)
  redis_provision_node_threshold_percent: 12 (%)
  max_simultaneous_backups: 4
  slave_ha: enabled
  slave_ha_grace_period: 600
  slave_ha_cooldown_period: 3600
  slave_ha_bdb_cooldown_period: 7200
  parallel_shards_upgrade: 0
  show_internals: disabled
  expose_hostnames_for_all_suffixes: disabled
  login_lockout_threshold: 5
  login_lockout_duration: 1800
  login_lockout_counter_reset_after: 900
  default_concurrent_restore_actions: 10
  endpoint_rebind_propagation_grace_time: 15
  data_internode_encryption: disabled
  redis_upgrade_policy: major
  db_conns_auditing: disabled
  watchdog profile: local-network
  http support: enabled
  upgrade mode: disabled
  cm_session_timeout_minutes: 15
  cm_port: 8443
  cnm_http_port: 8080
  cnm_https_port: 9443
  bigstore_driver: speedb
```

info db

Shows the current configuration for databases.

```
rladm info db [ {db:<id> | <name>} ]
```

Parameters

Parameter	Description
db:id	ID of the specified database (optional)
name	Name of the specified database (optional)

Returns

Returns the current configuration for all databases.

If db:<id> or <name> is specified, returns the current configuration for the specified database.

Example

```
$ rladm info db db:1
db:1 [database1]:
  client_buffer_limits: 1GB (hard limit)/512MB (soft limit) in 30 seconds
  slave_buffer: auto
  pubsub_buffer_limits: 32MB (hard limit)/8MB (soft limit) in 60 seconds
  proxy_client_buffer_limits: 0KB (hard limit)/0KB (soft limit) in 0 seconds
  proxy_slave_buffer_limits: 1GB (hard limit)/512MB (soft limit) in 60 seconds
  proxy_pubsub_buffer_limits: 32MB (hard limit)/8MB (soft limit) in 60 seconds
  repl_backlog: 1.02MB (1073741 bytes)
  repl_timeout: 360 seconds
  repl_diskless: default
  master_persistence: disabled
  maxclients: 10000
  conns: 5
  conns_type: per-thread
  sched_policy: cmp
  max_aof_file_size: 300GB
  max_aof_load_time: 3600 seconds
  dedicated_replicaof_threads: 5
  max_client_pipeline: 200
  max_shard_pipeline: 2000
  max_connections: 0
  oss_cluster: disabled
  oss_cluster_api_preferred_ip_type: internal
  gradual_src_mode: disabled
  gradual_src_max_sources: 1
  gradual_sync_mode: auto
  gradual_sync_max_shards_per_source: 1
  slave_ha: disabled (database)
  mkms: enabled
  oss_sharding: disabled
  mtls_allow_weak_hashing: disabled
  mtls_allow_outdated_certs: disabled
  data_internode_encryption: disabled
  proxy_policy: single
  db_conns_auditing: disabled
  syncer_mode: centralized
```

info node

Lists the current configuration for all nodes.

```
rladm info node [ <id> ]
```

Parameters

Parameter	Description
id	ID of the specified node

Returns

Returns the current configuration for all nodes.

If <id> is specified, returns the current configuration for the specified node.

Example

```
$ rladm info node 3
Command Output: node:3
  address: 198.51.100.17
  external addresses: N/A
  recovery path: N/A
  quorum only: disabled
  max redis servers: 100
  max listeners: 100
```

info proxy

Lists the current configuration for a proxy.

```
rladmin info proxy { <id> | all }
```

Parameters

Parameter	Description
id	ID of the specified proxy
all	Show the current configuration for all proxies (optional)

Returns

If no parameter is specified or the `all` option is specified, returns the current configuration for all proxies.

If `<id>` is specified, returns the current configuration for the specified proxy.

Example

```
$ rladm info proxy
proxy:1
  mode: dynamic
  scale_threshold: 80 (%)
  scale_duration: 30 (seconds)
  max_threads: 8
  threads: 3
```

Updated: August 17, 2023

rladmin migrate

Moves Redis Enterprise shards or endpoints to a new node in the same cluster.

migrate all_master_shards

Moves all primary shards of a specified database or node to a new node in the same cluster.

```
rladmin migrate { db { db:<id> | <name> } | node <origin node ID> }
  all_master_shards
  target_node <id>
  [ override_policy ]
```

Parameters

Parameter	Type/Value	Description
-----------	------------	-------------

Parameter	Type/Value	Description
db	db:<id> name	Limits migration to a specific database
node	integer	Limits migration to a specific origin node
target_node	integer	Migration target node
override_policy		Overrides the rack aware policy and allows primary and replica shards on the same node

Returns

Returns Done if the migration completed successfully. Otherwise, returns an error.

Use [rladmin status shards](#) to verify the migration completed.

Example

```
$ rladmin status shards db db:6 sort ROLE
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:6 tr02 redis:14 node:3 master 0-4095 3.01MB OK
db:6 tr02 redis:16 node:3 master 4096-8191 3.2MB OK
db:6 tr02 redis:18 node:3 master 8192-12287 3.2MB OK
db:6 tr02 redis:20 node:3 master 12288-16383 3.01MB OK
$ rladmin migrate db db:6 all_master_shards target_node 1
Monitoring 8b0f28e2-4342-427a-a8e3-a68cba653ffe
queued - migrate_shards
running - migrate_shards
Executing migrate_redis with shards_uids ['18', '14', '20', '16']
0completed - migrate_shards
Done
$ rladmin status shards node 1
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:6 tr02 redis:14 node:1 master 0-4095 3.22MB OK
db:6 tr02 redis:16 node:1 master 4096-8191 3.22MB OK
db:6 tr02 redis:18 node:1 master 8192-12287 3.22MB OK
db:6 tr02 redis:20 node:1 master 12288-16383 2.99MB OK
```

migrate all_shards

Moves all shards on a specified node to a new node in the same cluster.

```
rladmin migrate node <origin node ID>
  [ max_concurrent_bdb_migrations <value> ]
  all_shards
  target_node <id>
  [ override_policy ]
```

Parameters

Parameter	Type/Value	Description
node	integer	Limits migration to a specific origin node
max_concurrent_bdb_migrations	integer	Sets the maximum number of concurrent endpoint migrations
override_policy		Overrides the rack aware policy and allows primary and replica shards on the same node

Returns

Returns Done if the migration completed successfully. Otherwise, returns an error.

Use [rladmin status shards](#) to verify the migration completed.

Example

```
$ rladm status shards node 1
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:5 tr01 redis:12 node:1 master 0-16383 3.04MB OK
db:6 tr02 redis:15 node:1 slave 0-4095 2.93MB OK
db:6 tr02 redis:17 node:1 slave 4096-8191 2.93MB OK
db:6 tr02 redis:19 node:1 slave 8192-12287 3.08MB OK
db:6 tr02 redis:21 node:1 slave 12288-16383 3.08MB OK
$ rladm migrate node 1 all_shards target_node 2
Monitoring 71a4f371-9264-4398-a454-ce3ff4858c09
queued - migrate_shards
.running - migrate_shards
Executing migrate_redis with shards_uids ['21', '15', '17', '19']
Executing migrate_redis with shards_uids ['12']
Ocompleted - migrate_shards
Done
$ rladm status shards node 2
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:5 tr01 redis:12 node:2 master 0-16383 3.14MB OK
db:6 tr02 redis:15 node:2 slave 0-4095 2.96MB OK
db:6 tr02 redis:17 node:2 slave 4096-8191 2.96MB OK
db:6 tr02 redis:19 node:2 slave 8192-12287 2.96MB OK
db:6 tr02 redis:21 node:2 slave 12288-16383 2.96MB OK
```

migrate all_slave_shards

Moves all replica shards of a specified database or node to a new node in the same cluster.

```
rladmin migrate { db { db:<id> | <name> } | node <origin node ID> }
    all_slave_shards
    target_node <id>
    [ override_policy ]
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Limits migration to a specific database
node	integer	Limits migration to a specific origin node
target_node	integer	Migration target node
override_policy		Overrides the rack aware policy and allows primary and replica shards on the same node

Returns

Returns Done if the migration completed successfully. Otherwise, returns an error.

Use `rladmin status shards` to verify the migration completed.

Example

```
$ rladm status shards db db:6 node 2
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:6 tr02 redis:15 node:2 slave 0-4095 3.06MB OK
db:6 tr02 redis:17 node:2 slave 4096-8191 3.06MB OK
db:6 tr02 redis:19 node:2 slave 8192-12287 3.06MB OK
db:6 tr02 redis:21 node:2 slave 12288-16383 3.06MB OK
$ rladm migrate db db:6 all_slave_shards target_node 3
Monitoring 5d36a98c-3dc8-435f-8ed9-35809ba017a4
queued - migrate_shards
.running - migrate_shards
Executing migrate_redis with shards_uids ['15', '17', '21', '19']
0completed - migrate_shards
Done
$ rladm status shards db db:6 node 3
SHARDS:
DB:ID NAME ID NODE ROLE SLOTS USED_MEMORY STATUS
db:6 tr02 redis:15 node:3 slave 0-4095 3.04MB OK
db:6 tr02 redis:17 node:3 slave 4096-8191 3.04MB OK
db:6 tr02 redis:19 node:3 slave 8192-12287 3.04MB OK
db:6 tr02 redis:21 node:3 slave 12288-16383 3.04MB OK
```

migrate endpoint_to_shards

Moves database endpoints to the node where the majority of primary shards are located.

```
rladmin migrate [ db { db:<id> | <name> } ]
  endpoint_to_shards
  [ restrict_target_node <id> ]
  [ commit ]
  [ max_concurrent_bdb_migrations <value> ]
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Limits migration to a specific database
restrict_target_node	integer	Moves the endpoint only if the target node matches the specified node
commit		Performs endpoint movement
max_concurrent_bdb_migrations	integer	Sets the maximum number of concurrent endpoint migrations

Returns

Returns a list of steps to perform the migration. If the `commit` flag is set, the steps will run and return `Finished` successfully if they were completed. Otherwise, returns an error.

Use `rladmin status endpoints` to verify that the endpoints were moved.

Example

```

$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID    NAME    ID          NODE      ROLE           SSL
db:6     tr02   endpoint:6:1  node:3    all-master-shards  No
$ rladm migrate db db:6 endpoint_to_shards
* Going to bind endpoint:6:1 to node 1
Dry-run completed, add 'commit' argument to execute
$ rladm migrate db db:6 endpoint_to_shards commit
* Going to bind endpoint:6:1 to node 1
Executing bind endpoint:6:1: 000.
Finished successfully
$ rladm status endpoints db db:6
ENDPOINTS:
DB:ID    NAME    ID          NODE      ROLE           SSL
db:6     tr02   endpoint:6:1  node:1    all-master-shards  No

```

migrate shard

Moves one or more shards to a new node in the same cluster.

```

rladmin migrate shard <id1.. idN>
  [ preserve_roles ]
  target_node <id>
  [ override_policy ]

```

Parameters

Parameter	Type/Value	Description
shard	list of shard IDs	Shards to migrate
preserve_roles		Performs an additional failover to guarantee the primary shards' roles are preserved
target_node	integer	Migration target node
override_policy		Overrides the rack aware policy and allows primary and replica shards on the same node

Returns

Returns Done if the migration completed successfully. Otherwise, returns an error.

Use `rladmin status shards` to verify the migration completed.

Example

```

$ rladm status shards db db:5
SHARDS:
DB:ID    NAME    ID          NODE      ROLE      SLOTS      USED_MEMORY    STATUS
db:5     tr01   redis:12    node:2    master    0-16383   3.01MB    OK
db:5     tr01   redis:13    node:3    slave     0-16383   3.1MB     OK
$ rladm migrate shard 13 target_node 1
Monitoring d2637eea-9504-4e94-a70c-76df087efcb2
queued - migrate_shards
.running - migrate_shards
Executing migrate_redis with shards_uids ['13']
Ocompleted - migrate_shards
Done
$ rladm status shards db db:5
SHARDS:
DB:ID    NAME    ID          NODE      ROLE      SLOTS      USED_MEMORY    STATUS
db:5     tr01   redis:12    node:2    master    0-16383   3.01MB    OK
db:5     tr01   redis:13    node:1    slave     0-16383   3.04MB    OK

```

rladmin node

`rladmin node` commands manage nodes in the cluster.

Command	Description
<code>addr</code>	Sets a node's internal IP address.
<code>enslave</code>	Changes a node's resources to replicas.
<code>external_addr</code>	Configures a node's external IP addresses.
<code>maintenance_mode</code>	Turns quorum-only mode on or off for a node.
<code>recovery_path</code>	Sets a node's local recovery path.
<code>remove</code>	Removes a node from the cluster.
<code>snapshot</code>	Manages snapshots of the state of a node's resources.

Updated: June 10, 2022

rladmin node addr set

Sets the internal IP address of a node. You can only set the internal IP address when the node is down.

```
rladmin node <ID> addr set <IP address>
```

Parameters

Parameter	Type/Value	Description
<code>node</code>	integer	Sets the internal IP address of the specified node
<code>addr</code>	IP address	Sets the node's internal IP address to the specified IP address

Returns

Returns `Updated successfully` if the IP address was set. Otherwise, it returns an error.

Use `rladmin status nodes` to verify the internal IP address was changed.

Example

```

$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM VERSION
STATUS
*node:1 master 192.0.2.2 3d99db1fdf4b 5/100 6 16.06GB/19.54GB 12.46GB/16.02GB 6.2.12-
37 OK
node:2 slave 192.0.2.3 fc7a3d332458 0/100 6 -/19.54GB -/16.02GB 6.2.12-
37 DOWN, last seen 33s ago
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 16.06GB/19.54GB 12.46GB/16.02GB 6.2.12-
37 OK
$ rladm node 2 addr set 192.0.2.5
Updated successfully.
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM VERSION
STATUS
*node:1 master 192.0.2.2 3d99db1fdf4b 5/100 6 14.78GB/19.54GB 11.18GB/16.02GB 6.2.12-
37 OK
node:2 slave 192.0.2.5 fc7a3d332458 0/100 6 14.78GB/19.54GB 11.26GB/16.02GB 6.2.12-
37 OK
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 14.78GB/19.54GB 11.18GB/16.02GB 6.2.12-
37 OK

```

Updated: June 10, 2022

rladmin node enslave

Changes the resources of a node to replicas.

node enslave

Changes all of the node's endpoints and shards to replicas.

```

rladmin node <ID> enslave
  [demote_node]
  [retry_timeout_seconds <seconds>]

```

Parameters

Parameter	Type/Value	Description
node	integer	Changes all of the node's endpoints and shards to replicas
demote_node		If the node is a primary node, changes the node to replica
retry_timeout_seconds	integer	Retries on failure until the specified number of seconds has passed.

Returns

Returns OK if the roles were successfully changed. Otherwise, it returns an error.

Use `rladmin status shards` to verify that the roles were changed.

Example

```

$ rladm status shards node 2
SHARDS:
DB:ID      NAME        ID          NODE       ROLE        SLOTS           USED_MEMORY
STATUS
db:6       tr02        redis:14   node:2     master      0-4095          3.2MB
OK
db:6       tr02        redis:16   node:2     master      4096-8191         3.12MB
OK
db:6       tr02        redis:18   node:2     master      8192-12287        3.16MB
OK
db:6       tr02        redis:20   node:2     master      12288-16383        3.12MB
OK
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE      ADDRESS    EXTERNAL_ADDRESS  HOSTNAME      SHARDS CORES      FREE_RAM      PROVISIONAL_RAM
VERSION STATUS
*node:1 slave    192.0.2.12 198.51.100.1      3d99db1fdf4b 1/100  6          14.43GB/19.54GB 10.87GB/16.02GB
6.2.12-37 OK
node:2 master   192.0.2.13 198.51.100.2      fc7a3d332458 4/100  6          14.43GB/19.54GB 10.88GB/16.02GB
6.2.12-37 OK
node:3 slave    192.0.2.14                    b87cc06c830f  5/120  6          14.43GB/19.54GB 10.83GB/16.02GB
6.2.12-37 OK
$ rladm node 2 enslave demote_node
Performing enslave_node action on node:2: 100%
OK
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE      ADDRESS    EXTERNAL_ADDRESS  HOSTNAME      SHARDS CORES      FREE_RAM      PROVISIONAL_RAM
VERSION STATUS
*node:1 master   192.0.2.12 198.51.100.1      3d99db1fdf4b 1/100  6          14.72GB/19.54GB 10.91GB/16.02GB
6.2.12-37 OK
node:2 slave    192.0.2.13 198.51.100.2      fc7a3d332458 4/100  6          14.72GB/19.54GB 11.17GB/16.02GB
6.2.12-37 OK
node:3 slave    192.0.2.14                    b87cc06c830f  5/120  6          14.72GB/19.54GB 10.92GB/16.02GB
6.2.12-37 OK
$ rladm status shards node 2
SHARDS:
DB:ID      NAME        ID          NODE       ROLE        SLOTS           USED_MEMORY
STATUS
db:6       tr02        redis:14   node:2     slave       0-4095          2.99MB
OK
db:6       tr02        redis:16   node:2     slave       4096-8191         3.01MB
OK
db:6       tr02        redis:18   node:2     slave       8192-12287        2.93MB
OK
db:6       tr02        redis:20   node:2     slave       12288-16383        3.06MB
OK

```

node enslave endpoints_only

Changes the role for all endpoints on a node to replica.

```

rladmin node <ID> enslave endpoints_only
  [retry_timeout_seconds <seconds>]

```

Parameters

Parameter	Type/Value	Description
node	integer	Changes all of the node's endpoints to replicas
retry_timeout_seconds	integer	Retries on failure until the specified number of seconds has passed.

Returns

Returns OK if the roles were successfully changed. Otherwise, it returns an error.

Use `rladmin status endpoints` to verify that the roles were changed.

Example

```
$ rladm status endpoints
ENDPOINTS:
DB:ID      NAME      ID          NODE      ROLE
SSL
db:5       tr01      endpoint:5:1    node:1    single
No
db:6       tr02      endpoint:6:1    node:3    all-master-shards
No
$ rladm node 1 enslave endpoints_only
Performing enslave_node action on node:1: 100%
OK
$ rladm status endpoints
ENDPOINTS:
DB:ID      NAME      ID          NODE      ROLE
SSL
db:5       tr01      endpoint:5:1    node:3    single
No
db:6       tr02      endpoint:6:1    node:3    all-master-shards
No
```

node enslave shards_only

Changes the role for all shards of a node to replica.

```
rladmin node <ID> enslave shards_only
  [retry_timeout_seconds <seconds>]
```

Parameters

Parameter	Type/Value	Description
node	integer	Changes all of the node's shards to replicas
retry_timeout_seconds	integer	Retries on failure until the specified number of seconds has passed.

Returns

Returns OK if the roles were successfully changed. Otherwise, it returns an error.

Use `rladmin status shards` to verify that the roles were changed.

Example

```

$ rladm status shards node 3
SHARDS:
DB:ID      NAME        ID          NODE      ROLE       SLOTS           USED_MEMORY
STATUS
db:5        tr01        redis:12    node:3    master     0-16383        3.04MB
OK
db:6        tr02        redis:15    node:3    master     0-4095         4.13MB
OK
db:6        tr02        redis:17    node:3    master     4096-8191        4.13MB
OK
db:6        tr02        redis:19    node:3    master     8192-12287        4.13MB
OK
db:6        tr02        redis:21    node:3    master     12288-16383        4.13MB
OK
$ rladm node 3 enslave shards_only
Performing enslave_node action on node:3: 100%
OK
$ rladm status shards node 3
SHARDS:
DB:ID      NAME        ID          NODE      ROLE       SLOTS           USED_MEMORY
STATUS
db:5        tr01        redis:12    node:3    slave      0-16383        2.98MB
OK
db:6        tr02        redis:15    node:3    slave      0-4095         4.23MB
OK
db:6        tr02        redis:17    node:3    slave      4096-8191        4.11MB
OK
db:6        tr02        redis:19    node:3    slave      8192-12287        4.19MB
OK
db:6        tr02        redis:21    node:3    slave      12288-16383        4.27MB
OK

```

Updated: June 10, 2022

rladmin node external_addr

Configures a node's external IP addresses.

node external_addr add

Adds an external IP address that accepts inbound user connections for the node.

```

rladmin node <ID> external_addr
    add <IP address>

```

Parameters

Parameter	Type/Value	Description
node	integer	Adds an external IP address for the specified node
IP address	IP address	External IP address of the node

Returns

Returns Updated successfully if the IP address was added. Otherwise, it returns an error.

Use [rladmin status nodes](#) to verify the external IP address was added.

Example

```
$ rladm node 1 external_addr add 198.51.100.1
Updated successfully.
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.2 198.51.100.1 3d99db1fdf4b 5/100 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.3 fc7a3d332458 0/100 6 14.75GB/19.54GB 11.24GB/16.02GB
6.2.12-37 OK
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
```

node external_addr set

Sets one or more external IP addresses that accepts inbound user connections for the node.

```
rladmin node <ID> external_addr
    set <IP address 1> ... <IP address N>
```

Parameters

Parameter	Type/Value	Description
node	integer	Sets external IP addresses for the specified node
IP address	list of IP addresses	Sets specified IP addresses as external addresses

Returns

Returns `Updated successfully` if the IP addresses were set. Otherwise, it returns an error.

Use `rladmin status nodes` to verify the external IP address was set.

Example

```
$ rladm node 2 external_addr set 198.51.100.2 198.51.100.3
Updated successfully.
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.2 198.51.100.1 3d99db1fdf4b 5/100 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.3 198.51.100.2,198.51.100.3 fc7a3d332458 0/100 6 14.75GB/19.54GB 11.23GB/16.02GB
6.2.12-37 OK
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
```

node external_addr remove

Removes the specified external IP address from the node.

```
rladmin node <ID> external_addr
    remove <IP address>
```

Parameters

Parameter	Type/Value	Description
node	integer	Removes an external IP address for the specified node

Parameter	Type/Value	Description
IP address	IP address	Removes the specified IP address of the node

Returns

Returns Updated successfully if the IP address was removed. Otherwise, it returns an error.

Use `rladmin status nodes` to verify the external IP address was removed.

Example

```
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.2 198.51.100.1 3d99db1fdf4b 5/100 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.3 198.51.100.2,198.51.100.3 fc7a3d332458 0/100 6 14.75GB/19.54GB 11.23GB/16.02GB
6.2.12-37 OK
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 14.75GB/19.54GB 11.15GB/16.02GB
6.2.12-37 OK
$ rladm node 2 external_addr remove 198.51.100.3
Updated successfully.
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.2 198.51.100.1 3d99db1fdf4b 5/100 6 14.74GB/19.54GB 11.14GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.3 198.51.100.2 fc7a3d332458 0/100 6 14.74GB/19.54GB 11.22GB/16.02GB
6.2.12-37 OK
node:3 slave 192.0.2.4 b87cc06c830f 5/120 6 14.74GB/19.54GB 11.14GB/16.02GB
6.2.12-37 OK
```

Updated: June 10, 2022

rladmin node maintenance_mode

Configures [quorum-only mode](#) on a node.

node maintenance_mode on

Migrates shards out of the node and turns the node into a quorum node to prevent shards from returning to it.

```
rladmin node <ID> maintenance_mode on
  [ keep_slave_shards ]
  [ demote_node ]
  [ max_concurrent_actions <integer> ]
```

Parameters

Parameter	Type/Value	Description
node	integer	Turns the specified node into a quorum node
demote_node		If the node is a primary node, changes the node to replica
keep_slave_shards		Keeps replica shards in the node and demotes primary shards to replicas

Parameter	Type/Value	Description
max_concurrent_actions	integer	Maximum number of concurrent actions during node maintenance

Returns

Returns OK if the node was converted successfully. If the cluster does not have enough resources to migrate the shards, the process returns a warning.

Use `rladmin status nodes` to verify the node became a quorum node.

Example

```
$ rladm node 2 maintenance_mode on
Performing maintenance_on action on node:2: 0%
created snapshot NodeSnapshot<name=maintenance_mode_2022-05-12_20-25-37, time=None, node_uid=2>

node:2 will not accept any more shards
Performing maintenance_on action on node:2: 100%
OK
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.12 198.51.100.1 3d99db1fdf4b 5/100 6 14.21GB/19.54GB 10.62GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.13 198.51.100.2 fc7a3d332458 0/0 6 14.21GB/19.54GB 0KB/0KB
6.2.12-37 OK
node:4 slave 192.0.2.14 6d754fe12cb9 5/100 6 14.21GB/19.54GB 10.62GB/16.02GB
6.2.12-37 OK
```

node maintenance_mode off

Turns maintenance mode off and returns the node to its previous state.

```
rladmin node <ID> maintenance_mode off
[ { snapshot_name <name> | skip_shards_restore } ]
[ max_concurrent_actions <integer> ]
```

Parameters

Parameter	Type/Value	Description
node	integer	Restores the node back to the previous state
max_concurrent_actions	integer	Maximum number of concurrent actions during node maintenance
skip_shards_restore		Does not restore shards back to the node
snapshot_name	string	Restores the node back to a state stored in the specified snapshot

Returns

Returns OK if the node was restored successfully.

Use `rladmin status nodes` to verify the node was restored.

Example

```

$ rladm node 2 maintenance_mode off
Performing maintenance_off action on node:2: 0%
Found snapshot: NodeSnapshot<name=maintenance_mode_2022-05-12_20-25-37, time=2022-05-12T20:25:37Z, node_uid=2>
Performing maintenance_off action on node:2: 0%
migrate redis:12 to node:2: executing
Performing maintenance_off action on node:2: 0%
migrate redis:12 to node:2: finished
Performing maintenance_off action on node:2: 0%
migrate redis:17 to node:2: executing

migrate redis:15 to node:2: executing
Performing maintenance_off action on node:2: 0%
migrate redis:17 to node:2: finished

migrate redis:15 to node:2: finished
Performing maintenance_off action on node:2: 0%
failover redis:16: executing

failover redis:14: executing
Performing maintenance_off action on node:2: 0%
failover redis:16: finished

failover redis:14: finished
Performing maintenance_off action on node:2: 0%
failover redis:18: executing
Performing maintenance_off action on node:2: 0%
failover redis:18: finished

migrate redis:21 to node:2: executing

migrate redis:19 to node:2: executing
Performing maintenance_off action on node:2: 0%
migrate redis:21 to node:2: finished

migrate redis:19 to node:2: finished

failover redis:20: executing
Performing maintenance_off action on node:2: 0%
failover redis:20: finished
Performing maintenance_off action on node:2: 0%
rebind endpoint:6:1: executing
Performing maintenance_off action on node:2: 0%
rebind endpoint:6:1: finished
Performing maintenance_off action on node:2: 100%
OK
$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE      ADDRESS    EXTERNAL_ADDRESS  HOSTNAME        SHARDS CORES     FREE_RAM          PROVISIONAL_RAM
VERSION STATUS
*node:1 master    192.0.2.12 198.51.100.1    3d99db1fdf4b  5/100   6          14.2GB/19.54GB  10.61GB/16.02GB
6.2.12-37 OK
node:2 slave     192.0.2.13 198.51.100.2    fc7a3d332458  5/100   6          14.2GB/19.54GB  10.61GB/16.02GB
6.2.12-37 OK
node:4 slave     192.0.2.14           6d754fe12cb9  0/100   6          14.2GB/19.54GB  10.69GB/16.02GB
6.2.12-37 OK

```

Updated: June 10, 2022

rladmin node recovery_path set

Sets the node's local recovery path, which specifies the directory where [persistence files](#) are stored. You can use these persistence files to [recover a failed database](#).

```
rladmin node <ID> recovery_path set <path>
```

Parameters

Parameter	Type/Value	Description
node	integer	Sets the recovery path for the specified node
path	filepath	Path to the folder where persistence files are stored

Returns

Returns Updated successfully if the recovery path was set. Otherwise, it returns an error.

Example

```
$ rladmin node 2 recovery_path set /var/opt/redislabs/persist/redis
Updated successfully.
```

Updated: August 31, 2022

rladmin node remove

Removes the specified node from the cluster.

```
rladmin node <ID> remove [ wait_for_persistence { enabled | disabled } ]
```

Parameters

Parameter	Type/Value	Description
node	integer	The node to remove from the cluster
wait_for_persistence	enabled disabled	Ensures persistence files are available for recovery. The cluster policy persistent_node_removal determines the default value.

Returns

Returns OK if the node was removed successfully. Otherwise, it returns an error.

Use [rladmin status nodes](#) to verify that the node was removed.

Example

```

$ rladm status nodes
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.12 198.51.100.1 3d99db1fdf4b 5/100 6 14.26GB/19.54GB 10.67GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.13 198.51.100.2 fc7a3d332458 4/100 6 14.26GB/19.54GB 10.71GB/16.02GB
6.2.12-37 OK
node:3 slave 192.0.2.14 b87cc06c830f 1/120 6 14.26GB/19.54GB 10.7GB/16.02GB
6.2.12-37 OK
$ rladm node 3 remove
Performing remove action on node:3: 100%
OK
CLUSTER NODES:
NODE:ID ROLE ADDRESS EXTERNAL_ADDRESS HOSTNAME SHARDS CORES FREE_RAM PROVISIONAL_RAM
VERSION STATUS
*node:1 master 192.0.2.12 198.51.100.1 3d99db1fdf4b 5/100 6 14.34GB/19.54GB 10.74GB/16.02GB
6.2.12-37 OK
node:2 slave 192.0.2.13 198.51.100.2 fc7a3d332458 5/100 6 14.34GB/19.54GB 10.74GB/16.02GB
6.2.12-37 OK

```

Updated: March 7, 2023

rladmin node snapshot

Manages [snapshots](#) of the state of a node's shards and endpoints.

node snapshot create

Creates a snapshot of a node's current state.

```
rladmin node <ID> snapshot create <name>
```

Parameters

Parameter	Type/Value	Description
node	integer	Creates a snapshot of the specified node
name	string	Name of the created snapshot

Returns

Returns Done if the snapshot was created successfully. Otherwise, returns an error.

Example

```
$ rladm node 1 snapshot create snap1
Creating node snapshot 'snap1' for node:1
Done.
```

node snapshot delete

Deletes an existing snapshot of a node.

```
rladmin node <ID> snapshot delete <name>
```

Parameters

Parameter	Type/Value	Description
node	integer	Deletes a snapshot of the specified node
name	string	Deletes the specified snapshot

Returns

Returns Done if the snapshot was deleted successfully. Otherwise, returns an error.

Example

```
$ rladm node 1 snapshot delete snap1
Deleting node snapshot 'snap1' for node:1
Done.
```

node snapshot list

Displays a list of created snapshots for the specified node.

```
rladmin node <ID> snapshot list
```

Parameters

Parameter	Type/Value	Description
node	integer	Displays snapshots of the specified node

Returns

Returns a list of snapshots of the specified node.

Example

```
$ rladm node 2 snapshot list
Name          Node    Time
snap2         2       2022-05-12T19:27:51Z
```

node snapshot restore

Restores a node as close to the stored snapshot as possible.

```
rladmin node <ID> snapshot restore <name>
```

Parameters

Parameter	Type/Value	Description
node	integer	Restore the specified node from a snapshot.
restore	string	Name of the snapshot used to restore the node.

Returns

Returns Snapshot restore completed successfully if the actions needed to restore the snapshot completed successfully. Otherwise, it returns an error.

Example

```

$ rladm node 2 snapshot restore snap2
Reading node snapshot 'snap2' for node:2
Planning restore
Planned actions:
* migrate redis:15 to node:2
* failover redis:14
* migrate redis:17 to node:2
* failover redis:16
* migrate redis:19 to node:2
* failover redis:18
* migrate redis:21 to node:2
* failover redis:20
Proceed?[Y]es/[N]o? Y
2022-05-12T19:43:31.486613 Scheduling 8 actions
[2022-05-12T19:43:31.521422 Actions Status: 8 waiting ]
* [migrate redis:21 to node:2] waiting => executing
* [migrate redis:19 to node:2] waiting => executing
* [migrate redis:17 to node:2] waiting => executing
* [migrate redis:15 to node:2] waiting => executing
[2022-05-12T19:43:32.586084 Actions Status: 4 executing | 4 waiting ]
* [migrate redis:21 to node:2] executing => finished
* [migrate redis:19 to node:2] executing => finished
* [migrate redis:17 to node:2] executing => finished
* [migrate redis:15 to node:2] executing => finished
* [failover redis:20] waiting => executing
* [failover redis:18] waiting => executing
* [failover redis:16] waiting => executing
* [failover redis:14] waiting => executing
[2022-05-12T19:43:33.719496 Actions Status: 4 finished | 4 executing ]
* [failover redis:20] executing => finished
* [failover redis:18] executing => finished
* [failover redis:16] executing => finished
* [failover redis:14] executing => finished
Snapshot restore completed successfully.

```

Updated: June 10, 2022

rladmin placement

Configures the shard placement policy for a specified database.

```

rladmin placement
  db { db:<id> | <name> }
    { dense | sparse }

```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Configures shard placement for the specified database
dense		Places new shards on the same node as long as it has resources
sparse		Places new shards on the maximum number of available nodes within the cluster

Returns

Returns the new shard placement policy if the policy was changed successfully. Otherwise, it returns an error.

Use `rladmin status databases` to verify that the failover completed.

Example

```
$ rladmin status databases
DATABASES:
DB:ID NAME      TYPE   STATUS  SHARDS  PLACEMENT    REPLICATION  PERSISTENCE  ENDPOINT
db:5  tr01      redis  active   1        dense       enabled      aof          redis-
12000.cluster.local:12000
$ rladmin placement db db:5 sparse
Shards placement policy is now sparse
$ rladmin status databases
DATABASES:
DB:ID NAME      TYPE   STATUS  SHARDS  PLACEMENT    REPLICATION  PERSISTENCE  ENDPOINT
db:5  tr01      redis  active   1        sparse      enabled      aof          redis-
12000.cluster.local:12000
```

Updated: June 10, 2022

rladmin recover

Recover databases in recovery mode.

recover all

Recover all databases in recovery mode.

```
rladmin recover all
  [ only_configuration ]
```

Parameters

Parameters	Type/Value	Description
only_configuration		Recover database configuration without data

Returns

Returns Completed successfully if the database was recovered. Otherwise, returns an error.

recover db

Recover a specific database in recovery mode.

```
rladmin recover db { db:<id> | <name> }
  [ only_configuration ]
```

Parameters

Parameters	Type/Value	Description
db	db:<id> name	Database to recover
only_configuration		Recover database configuration without data

Returns

Returns Completed successfully if the database was recovered. Otherwise, returns an error.

recover list

Shows a list of all databases that are currently in recovery mode.

```
rladmin recover list
```

Parameters

None

Returns

Displays a list of all recoverable databases. If no databases are in recovery mode, returns No recoverable databases found.

Example

```
$ rladm recover list
DATABASES IN RECOVERY STATE:
DB:ID  NAME   TYPE    SHARDS  REPLICATION  PERSISTENCE  STATUS
db:5   tr01   redis   1        enabled       aof          missing-files
db:6   tr02   redis   4        enabled       snapshot     ready
```

recover s3_import

Imports current database snapshot files from an AWS S3 bucket to a directory on the node.

```
$ rladm recover s3_import
      s3_bucket <bucket name>
      [ s3_prefix <prefix> ]
      s3_access_key_id <access key>
      s3_secret_access_key <secret access key>
      local_path <path>
```

Parameters

Parameters	Type/Value	Description
s3_bucket	string	S3 bucket name
s3_prefix	string	S3 object prefix
s3_access_key_id	string	S3 access key ID
s3_secret_access_key	string	S3 secret access key
local_path	filepath	Local import path where all database snapshots will be imported

Returns

Returns Completed successfully if the database files were imported. Otherwise, returns an error.

Updated: June 10, 2022

rladmin restart

Schedules a restart of the Redis Enterprise Software processes on primary and replica instances of a specific database.

```
rladmin restart db { db:<id> | <name> }
      [preserve_roles]
      [discard_data]
      [force_discard]
```

Parameters

Parameter	Type/Value	Description
db	db:<id> name	Restarts Redis Enterprise Software processes for the specified database
discard_data		Allows discarding data if there is no persistence or replication
force_discard		Forcibly discards data even if there is persistence or replication
preserve_roles		Performs an additional failover to maintain shard roles

Returns

Returns Done if the restart completed successfully. Otherwise, it returns an error.

Example

```
$ rladm restart db db:5 preserve_roles
Monitoring 1db07491-35da-4bb6-9bc1-56949f4c312a
active - SMUpgradeBDB init
active - SMUpgradeBDB stop_forwarding
active - SMUpgradeBDB stop_active_expire
active - SMUpgradeBDB check_slave
oactive - SMUpgradeBDB stop_active_expire
active - SMUpgradeBDB second_failover
completed - SMUpgradeBDB
Done
```

Updated: June 10, 2022

rladmin status

Displays the current cluster status and topology information.

status

Displays the current status of all nodes, databases, database endpoints, and shards on the cluster.

```
rladmin status
  [ extra <parameter> ]
  [ issues_only ]
```

Parameters

Parameter	Description
extra <parameter>	Extra options that show more information
issues_only	Filters out all items that have an OK status

Extra parameter	Description
extra all	Shows all extra information
extra backups	Shows periodic backup status
extra frag	Shows fragmented memory available after the restart
extra nodestats	Shows shards per node
extra rack_id	Shows rack_id if customer is not rack_aware
extra redis_version	Shows Redis version of all databases in the cluster

Extra parameter	Description
extra state_machine	Shows execution of state machine information
extra watchdog	Shows watchdog status

Returns

Returns tables of the status of all nodes, databases, and database endpoints on the cluster.

If issues_only is specified, it only shows instances that do not have an OK status.

Example

```
$ rladm status extra all
CLUSTER:
OK. Cluster master: 1 (198.51.100.2)
Cluster health: OK, [1, 0.1333333333333333, 0.0333333333333333]
failures/minute - avg1 1.00, avg15 0.13, avg60 0.03.

CLUSTER NODES:
NODE:ID ROLE      ADDRESS      EXTERNAL_ADDRESS HOSTNAME      MASTERS SLAVES OVERBOOKING_DEPTH SHARDS CORES
FREE_RAM      PROVISIONAL_RAM VERSION      SHA      RACK-ID STATUS
node:1 master 198.51.100.2          3d99db1fdf4b 4      0      10.91GB      4/100  6
14.91GB/19.54GB 10.91GB/16.02GB 6.2.12-37 5c2106 -      OK
node:2 slave 198.51.100.3          fc7a3d332458 0      0      11.4GB       0/100  6
14.91GB/19.54GB 11.4GB/16.02GB 6.2.12-37 5c2106 -      OK
*node:3 slave 198.51.100.4          b87cc06c830f 0      0      11.4GB       0/100  6
14.91GB/19.54GB 11.4GB/16.02GB 6.2.12-37 5c2106 -      OK

DATABASES:
DB:ID NAME      TYPE      STATUS SHARDS PLACEMENT REPLICATION PERSISTENCE ENDPOINT
EXEC_STATE EXEC_MACHINE BACKUP_PROGRESS MISSING_BACKUP_TIME REDIS_VERSION
db:3 database3 redis active 4      dense      disabled      disabled      redis-11103.cluster.local:11103 N/A
N/A           N/A           N/A           6.0.16

ENDPOINTS:
DB:ID      NAME      ID      NODE      ROLE      SSL      WATCHDOG_STATUS
db:3      database3 endpoint:3:1 node:1      single    No      OK

SHARDS:
DB:ID NAME      ID      NODE      ROLE      SLOTS      USED_MEMORY BACKUP_PROGRESS RAM_FRAG WATCHDOG_STATUS
STATUS
db:3 database3 redis:4 node:1 master 0-4095      2.08MB      N/A      4.73MB      OK
db:3 database3 redis:5 node:1 master 4096-8191    2.08MB      N/A      4.62MB      OK
db:3 database3 redis:6 node:1 master 8192-12287   2.08MB      N/A      4.59MB      OK
db:3 database3 redis:7 node:1 master 12288-16383   2.08MB      N/A      4.66MB      OK
```

status databases

Displays the current status of all databases on the cluster.

```
rladm status databases
[ extra <parameters> ]
[ sort <column_titles> ]
[ issues_only ]
```

Parameters

Parameter	Description
extra <parameter>	Extra options that show more information
sort <column_titles>	Sort results by specified column titles
issues_only	Filters out all items that have an OK status

Extra parameter	Description
extra all	Shows all extra information
extra backups	Shows periodic backup status
extra frag	Shows fragmented memory available after the restart
extra nodestats	Shows shards per node
extra rack_id	Shows rack_id if customer is not rack_aware
extra redis_version	Shows Redis version of all databases in the cluster
extra state_machine	Shows execution of state machine information
extra watchdog	Shows watchdog status

Returns

Returns a table of the status of all databases on the cluster.

If sort <column_titles> is specified, the result is sorted by the specified table columns.

If issues_only is specified, it only shows databases that do not have an OK status.

Example

```
$ rladmin status databases sort REPLICATION PERSISTENCE
DB:ID NAME      TYPE STATUS SHARDS PLACEMENT REPLICATION PERSISTENCE ENDPOINT
db:1 database1  redis active 1      dense     disabled    disabled   redis-
10269.testdbd11169.localhost:10269
db:2 database2  redis active 1      dense     disabled    snapshot   redis-
13897.testdbd11169.localhost:13897
db:3 database3  redis active 1      dense     enabled     snapshot   redis-
19416.testdbd13186.localhost:19416
```

status endpoints

Displays the current status of all endpoints on the cluster.

```
rladmin status endpoints
[ node <id> ]
[ extra <parameters> ]
[ sort <column_titles> ]
[ issues_only ]
```

Parameters

Parameter	Description
node <id>	Only show endpoints for the specified node ID
extra <parameter>	Extra options that show more information
sort <column_titles>	Sort results by specified column titles
issues_only	Filters out all items that have an OK status

Extra parameter	Description
extra all	Shows all extra information
extra backups	Shows periodic backup status
extra frag	Shows fragmented memory available after the restart
extra nodestats	Shows shards per node
extra rack_id	Shows rack_id if customer is not rack_aware
extra redis_version	Shows Redis version of all endpoints in the cluster
extra state_machine	Shows execution of state machine information

Extra parameter	Description
extra watchdog	Shows watchdog status

Returns

Returns a table of the status of all endpoints on the cluster.

If `sort <column_titles>` is specified, the result is sorted by the specified table columns.

If `issues_only` is specified, it only shows endpoints that do not have an OK status.

Example

```
$ rladmin status endpoints
DB:ID      NAME          ID          NODE      ROLE      SSL
db:1       database1    endpoint:1:1  node:1    single    No
db:2       database2    endpoint:2:1  node:2    single    No
db:3       database3    endpoint:3:1  node:3    single    No
```

status nodes

Displays the current status of all nodes on the cluster.

```
rladmin status nodes
[ extra <parameters> ]
[ sort <column_titles> ]
[ issues_only ]
```

Parameters

Parameter	Description
<code>extra <parameter></code>	Extra options that show more information
<code>sort <column_titles></code>	Sort results by specified column titles
<code>issues_only</code>	Filters out all items that have an OK status

Extra parameter	Description
<code>extra all</code>	Shows all extra information
<code>extra backups</code>	Shows periodic backup status
<code>extra frag</code>	Shows fragmented memory available after the restart
<code>extra nodestats</code>	Shows shards per node
<code>extra rack_id</code>	Shows <code>rack_id</code> if customer is not <code>rack_aware</code>
<code>extra redis_version</code>	Shows Redis version of all nodes in the cluster
<code>extra state_machine</code>	Shows execution of state machine information
<code>extra watchdog</code>	Shows watchdog status

Returns

Returns a table of the status of all nodes on the cluster.

If `sort <column_titles>` is specified, the result is sorted by the specified table columns.

If `issues_only` is specified, it only shows nodes that do not have an OK status.

Example

\$ rladm status nodes sort PROVISIONAL_RAM HOSTNAME						
CLUSTER NODES:		ADDRESS	EXTERNAL_ADDRESS	HOSTNAME	SHARDS	CORES
NODE_ID	ROLE	PROVISIONAL_RAM	VERSION	STATUS		
node:1	master	198.51.100.2 14.74GB/19.54GB	10.73GB/16.02GB	6.2.12-37	OK 3d99db1fdf4b	4/100 6
*node:3	slave	198.51.100.4 14.74GB/19.54GB	11.22GB/16.02GB	6.2.12-37	OK b87cc06c830f	0/100 6
node:2	slave	198.51.100.3 14.74GB/19.54GB	11.22GB/16.02GB	6.2.12-37	OK fc7a3d332458	0/100 6

status shards

Displays the current status of all shards on the cluster.

```
rladmin status shards
[ node <id> ]
[ db {db:<id> | <name>} ]
[ extra <parameters> ]
[ sort <column_titles> ]
[ issues_only ]
```

Parameters

Parameter	Description
node <id>	Only show shards for the specified node ID
db db:<id>	Only show shards for the specified database ID
db <name>	Only show shards for the specified database name
extra <parameter>	Extra options that show more information
sort <column_titles>	Sort results by specified column titles
issues_only	Filters out all items that have an OK status

Extra parameter	Description
extra all	Shows all extra information
extra backups	Shows periodic backup status
extra frag	Shows fragmented memory available after the restart
extra shardstats	Shows shards per node
extra rack_id	Shows rack_id if customer is not rack_aware
extra redis_version	Shows Redis version of all shards in the cluster
extra state_machine	Shows execution of state machine information
extra watchdog	Shows watchdog status

Returns

Returns a table of the status of all shards on the cluster.

If `sort <column_titles>` is specified, the result is sorted by the specified table columns.

If `issues_only` is specified, it only shows shards that do not have an OK status.

Example

```
$ rladm status shards sort USED_MEMORY ID
SHARDS:
DB:ID          NAME           ID          NODE        ROLE
SLOTS          USED_MEMORY
db:3           database3     redis:6    node:1      master
8192-12287    2.04MB        OK
db:3           database3     redis:4    node:1      master
0-4095         2.08MB        OK
db:3           database3     redis:5    node:1      master
4096-8191     2.08MB        OK
db:3           database3     redis:7    node:1      master
12288-16383   2.08MB        OK
```

Updated: June 10, 2022

rladmin suffix

Manages the DNS suffixes in the cluster.

suffix add

Adds a DNS suffix to the cluster.

```
rladmin suffix add name <name>
  [default]
  [internal]
  [mdns]
  [use_aaaa_ns]
  [slaves <ip>...]
```

Parameters

Parameter	Type/Value	Description
name	string	DNS suffix to add to the cluster
default		Sets the given suffix as the default. If a default already exists, this overwrites it.
internal		Forces the given suffix to use private IPs
mdns		Activates multicast DNS support for the given suffix
slaves	list of IPv4 addresses	The given suffix will notify the frontend DNS servers when a change in the frontend DNS has occurred
use_aaaa_ns		Activates IPv6 address support

Returns

Returns Added `suffixes` successfully if the suffix was added. Otherwise, it returns an error.

Example

```
$ rladm suffix add name new.rediscluster.local
Added suffixes successfully
```

suffix delete

Deletes an existing DNS suffix from the cluster.

```
rladmin suffix delete name <name>
```

Parameters

Parameter	Type/Value	Description
name	string	DNS suffix to delete from the cluster

Returns

Returns Suffix deleted successfully if the suffix was deleted. Otherwise, it returns an error.

Example

```
$ rladmin suffix delete name new.rediscluster.local
Suffix deleted successfully
```

suffix list

Lists the DNS suffixes in the cluster.

```
rladmin suffix list
```

Parameters

None

Returns

Returns a list of the DNS suffixes.

Example

```
$ rladmin suffix list
List of all suffixes:
cluster.local
new.rediscluster.local
```

Updated: June 10, 2022

rladmin tune

Configures parameters for databases, proxies, nodes, and clusters.

tune cluster

Configures cluster parameters.

```

rladmin tune cluster
[ repl_diskless { enabled | disabled } ]
[ redis_provision_node_threshold <size> ]
[ redis_migrate_node_threshold <size> ]
[ redis_provision_node_threshold_percent <percent> ]
[ redis_migrate_node_threshold_percent <percent> ]
[ max_simultaneous_backups <number> ]
[ failure_detection_sensitivity { high | low } ]
[ watchdog_profile { cloud | local-network } ]
[ slave_ha { enabled | disabled } ]
[ slave_ha_grace_period <seconds> ]
[ slave_ha_cooldown_period <seconds> ]
[ slave_ha_bdb_cooldown_period <seconds> ]
[ max_saved_events_per_type <value> ]
[ parallel_shards_upgrade <value> ]
[ default_concurrent_restore_actions <value> ]
[ show_internals { enabled | disabled } ]
[ expose_hostnames_for_all_suffixes { enabled | disabled } ]
[ redis_upgrade_policy { latest | major } ]
[ default_redis_version <value> ]
[ data_internode_encryption { enabled | disabled } ]
[ db_conns_auditing { enabled | disabled } ]
[ acl_pubsub_default { resetchannels | allchannels } ]
[ resp3_default { enabled | disabled } ]

```

Parameters

Parameters	Type/Value	Description
acl_pubsub_default	resetchannels allchannels	Default pub/sub ACL rule for all databases in the cluster: <ul style="list-style-type: none"> • resetchannels blocks access to all channels (restrictive) • allchannels allows access to all channels (permissive)
data_internode_encryption	enabled disabled	Activates or deactivates internode encryption for new databases
db_conns_auditing	enabled disabled	Activates or deactivates connection auditing by default for new databases of a cluster
default_concurrent_restore_actions	integer all	Default number of concurrent actions when restoring a node from a snapshot (positive integer or "all")
default_redis_version	version number	The default Redis database compatibility version used to create new databases.
expose_hostnames_for_all_suffixes	enabled disabled	The value parameter should be a version number in the form of "x.y" where x represents the major version number and y represents the minor version number. The final value corresponds to the desired version of Redis. You cannot set <code>default_redis_version</code> to a value higher than that supported by the current <code>redis_upgrade_policy</code> value.
failure_detection_sensitivity	high low	Exposes hostnames for all DNS suffixes Predefined thresholds and timeouts for failure detection (previously known as <code>watchdog_profile</code>) <ul style="list-style-type: none"> • high (previously <code>local-network</code>) – high failure detection sensitivity, lower thresholds, faster failure detection and failover • low (previously <code>cloud</code>) – low failure detection sensitivity, higher tolerance for latency variance (also called network jitter)

Parameters	Type/Value	Description
login_lockout_counter_reset_after	time in seconds	Time after failed login attempt before the counter resets to 0
login_lockout_duration	time in seconds	Time a locked account remains locked ("0" means only an admin can unlock the account)
login_lockout_threshold	integer	Number of failed sign-in attempts to trigger locking a user account ("0" means never lock the account)
max_saved_events_per_type	integer	Maximum number of events each type saved in CCS per object type
max_simultaneous_backups	integer (default: 4)	Number of database backups allowed to run at the same time. Combines with max_redis_forks (set by tune node) to determine the number of shard backups allowed to run simultaneously.
parallel_shards_upgrade	integer all	Number of shards upgraded in parallel during DB upgrade (positive integer or "all")
redis_migrate_node_threshold	size in MB	Memory (in MBs by default or can be specified) needed to migrate a database between nodes
redis_migrate_node_threshold_percent	percentage	Memory (in percentage) needed to migrate a database between nodes
redis_provision_node_threshold	size in MB	Memory (in MBs by default or can be specified) needed to provision a new database
redis_provision_node_threshold_percent	percentage	Memory (in percentage) needed to provision a new database
redis_upgrade_policy	latest major	<p>When you upgrade or create a new Redis database, this policy determines which version of Redis database compatibility is used.</p> <p>Supported values are:</p> <ul style="list-style-type: none"> • latest, which applies the most recent Redis compatibility update (<i>effective default prior to v6.2.4</i>) • major, which applies the most recent major release compatibility update (<i>default as of v6.2.4</i>).
repl_diskless	enabled disabled	Activates or deactivates diskless replication (can be overridden per database)
resp3_default	enabled disabled	Determines the default value of the resp3 option upon upgrading a database to version 7.2 (defaults to enabled)
show_internals	enabled disabled	Controls the visibility of internal databases that are only used for the cluster's management
slave_ha	enabled disabled	Activates or deactivates replica high availability in the cluster (enabled by default; use rladmin tune db to change slave_ha for a specific database)
slave_ha_bdb_cooldown_period	time in seconds (default: 7200)	Time (in seconds) a database must wait after its shards are relocated by replica high availability before it can go through another shard migration if another node fails (default is 2 hours)
slave_ha_cooldown_period	time in seconds (default: 3600)	Time (in seconds) replica high availability must wait after relocating shards due to node failure before performing another shard migration for any database in the cluster (default is 1 hour)
slave_ha_grace_period	time in seconds (default: 600)	Time (in seconds) between when a node fails and when replica high availability starts relocating shards to another node

Parameters	Type/Value	Description
watchdog_profile	cloud local-network	Watchdog profiles with preconfigured thresholds and timeouts (deprecated as of Redis Enterprise Software v6.4.2-69; use <code>failure_detection_sensitivity</code> instead) <ul style="list-style-type: none"> • <code>cloud</code> is suitable for common cloud environments and has a higher tolerance for latency variance (also called network jitter). • <code>local-network</code> is suitable for dedicated LANs and has better failure detection and failover times.

Returns

Returns `Finished` successfully if the cluster configuration was changed. Otherwise, it returns an error.

Use `radmin info cluster` to verify the cluster configuration was changed.

Example

```
$ radmin tune cluster slave_ha enabled
Finished successfully
$ radmin info cluster | grep slave_ha
slave_ha: enabled
```

tune db

Configures database parameters.

```
radmin tune db { db:<id> | <name> }
[ slave_buffer <valueMG | hard:soft:time> ]
[ client_buffer <value> ]
[ repl_backlog <valueMB | auto> ]
[ crdt_repl_backlog <valueMB | auto> ]
[ repl_timeout <seconds> ]
[ repl_diskless { enabled | disabled | default } ]
[ master_persistence { enabled | disabled } ]
[ maxclients <value> ]
[ schedpolicy { cmp | mru | spread | mnp } ]
[ max_shard_pipeline <value> ]
[ conns <value> ]
[ conns_type <value> ]
[ max_client_pipeline <value> ]
[ max_connections <value> ]
[ max_aof_file_size <size> ]
[ max_aof_load_time <seconds> ]
[ oss_cluster { enabled | disabled } ]
[ oss_cluster_api_preferred_ip_type <value> ]
[ slave_ha { enabled | disabled } ]
[ slave_ha_priority <value> ]
[ skip_import_analyze { enabled | disabled } ]
[ mkms { enabled | disabled } ]
[ continue_on_error ]
[ gradual_src_mode { enabled | disabled } ]
[ gradual_sync_mode { enabled | disabled | auto } ]
[ gradual_sync_max_shards_per_source <value> ]
[ module_name <value> ] [ module_config_params <value> ]
[ crdt_xadd_id_uniqueness_mode { liberal | semi-strict | strict } ]
[ metrics_export_all { enabled | disabled } ]
[ syncer_mode { distributed | centralized } ]
[ syncer_monitoring { enabled | disabled } ]
[ mtls_allow_weak_hashing { enabled | disabled } ]
[ mtls_allow_outdated_cert { enabled | disabled } ]
[ data_internode_encryption { enabled | disabled } ]
[ db_conns_auditing { enabled | disabled } ]
[ resp3 { enabled | disabled } ]
```

Parameters

Parameter	Type/Value	Description
db:id	integer	ID of the specified database
name	string	Name of the specified database
client_buffer	value in MB hard:soft:time	Redis client output buffer limits
conns	integer	Size of internal connection pool, specified per-thread or per-shard depending on conns_type
conns_type	per-thread per-shard	Specifies connection pool size as either per-thread or per-shard
continue_on_error		Flag that skips tuning shards that can't be reached
crdt_repl_backlog	value in MB auto	Size of the Active-Active replication buffer
crdt_xadd_id_uniqueness_mode	liberal semi-strict strict	XADD's behavior in an Active-Active database, defined as liberal, semi-strict, or strict (see descriptions below)
data_internode_encryption	enabled disabled	Activates or deactivates internode encryption for the database
db_conns_auditing	enabled disabled	Activates or deactivates database connection auditing for a database
gradual_src_mode	enabled disabled	Activates or deactivates gradual sync of sources
gradual_sync_max_shards_per_source	integer	Number of shards per sync source that can be replicated in parallel (positive integer)
gradual_sync_mode	enabled disabled auto	Activates, deactivates, or automatically determines gradual sync of source shards
master_persistence	enabled disabled	Activates or deactivates persistence of the primary shard
max_aof_file_size	size in MB	Maximum size (in MB, if not specified) of AOF file (minimum value is 10 GB)
max_aof_load_time	time in seconds	Time limit in seconds to load a shard from an append-only file (AOF). If exceeded, an AOF rewrite is initiated to decrease future load time. Minimum: 2700 seconds (45 minutes) Default: 3600 seconds (1 hour)
max_client_pipeline	integer	Maximum commands in the proxy's pipeline per client connection (max value is 2047, default value is 200)
max_connections	integer	Maximum client connections to the database's endpoint (default value is 0, which is unlimited)
max_shard_pipeline	integer	Maximum commands in the proxy's pipeline per shard connection (default value is 200)
maxclients	integer	Controls the maximum client connections between the proxy and shards (default value is 10000)
metrics_export_all	enabled disabled	Activates the exporter to expose all shard metrics
mkms	enabled disabled	Activates multi-key multi-slot commands
module_config_params	string	Configures module arguments at runtime. Enclose module_config_params within quotation marks.
module_name	search ReJSON graph timeseries bf rg	The module to configure with module_config_params

Parameter	Type/Value	Description
mtls_allow_outdated_cert	enabled disabled	Activates outdated certificates in mTLS connections
mtls_allow_weak_hashing	enabled disabled	Activates weak hashing (less than 2048 bits) in mTLS connections
oss_cluster	enabled disabled	Activates OSS cluster API
oss_cluster_api_preferred_ip_type	internal external	IP type for the endpoint and database in the OSS cluster API (default is internal)
repl_backlog	size in MB auto	Size of the replication buffer
repl_diskless	enabled disabled default	Activates or deactivates diskless replication (defaults to the cluster setting)
repl_timeout	time in seconds	Replication timeout (in seconds)
resp3	enabled disabled	Enables or deactivates RESP3 support (defaults to enabled)
schedpolicy	cmp mru spread mnp	Controls how server-side connections are used when forwarding traffic to shards
skip_import_analyze	enabled disabled	Skips the analyzing step when importing a database
slave_buffer	value in MB hard:soft:time	Redis replica output buffer limits
slave_ha	enabled disabled	Activates or deactivates replica high availability (defaults to the cluster setting)
slave_ha_priority	integer	Priority of the database in the replica high-availability mechanism
syncer_mode	distributed centralized	Configures syncer to run in distributed or centralized mode. For distributed syncer, the DMC policy must be all-nodes or all-master-nodes
syncer_monitoring	enabled disabled	Activates syncer monitoring

XADD behavior mode	Description
liberal	XADD succeeds with any valid ID (not recommended, allows duplicate IDs)
semi-strict	Allows a full ID. Partial IDs are completed with the unique database instance ID (not recommended, allows duplicate IDs).
strict	XADD fails if a full ID is given. Partial IDs are completed using the unique database instance ID.

Returns

Returns `Finished successfully` if the database configuration was changed. Otherwise, it returns an error.

Use `rladmin info db` to verify the database configuration was changed.

Example

```
$ rladm tune db db:4 repl_timeout 300
Tuning database: o
Finished successfully
$ rladm info db db:4 | grep repl_timeout
  repl_timeout: 300 seconds
```

tune node

Configures node parameters.

```
tune node { <id> | all }
  [ max_listeners <value> ]
  [ max_redis_forks <value> ]
  [ max_redis_servers <value> ]
  [ max_slave_full_syncs <value> ]
  [ quorum_only { enabled | disabled } ]
```

Parameters

Parameter	Type/Value	Description
id	integer	ID of the specified node
all		Configures settings for all nodes
max_listeners	integer	Maximum number of endpoints that may be bound to the node
max_redis_forks	integer	Maximum number of background processes forked from shards that may exist on the node at any given time
max_redis_servers	integer	Maximum number of shards allowed to reside on the node
max_slave_full_syncs	integer	Maximum number of simultaneous replica full-syncs that may be running at any given time (0: Unlimited, -1: Use cluster settings)
quorum_only	enabled disabled	If activated, configures the node as a quorum-only node

Returns

Returns `Finished successfully` if the node configuration was changed. Otherwise, it returns an error.

Use `rladmin info node` to verify the node configuration was changed.

Example

```
$ rladmin tune node 3 max_redis_servers 120
Finished successfully
$ rladmin info node 3 | grep "max redis servers"
  max redis servers: 120
```

tune proxy

Configures proxy parameters.

```
rladmin tune proxy { <id> | all }
  [ mode { static | dynamic } ]
  [ threads <value> ]
  [ max_threads <value> ]
  [ scale_threshold <value> ]
  [ scale_duration <seconds> ]
```

Parameters

Parameter	Type/Value	Description
id	integer	ID of the specified proxy
all		Configures settings for all proxies
max_threads	integer	Maximum number of threads allowed
mode	static dynamic	Determines if the proxy automatically adjusts the number of threads based on load size

Parameter	Type/Value	Description
scale_duration	time in seconds	Time of scale_threshold CPU utilization before the automatic proxy automatically scales
scale_threshold	percentage	CPU utilization threshold that triggers spawning new threads
threads	integer	Initial number of threads created at startup

Returns

Returns OK if the proxy configuration was changed. Otherwise, it returns an error.

Use [rladmin info proxy](#) to verify the proxy configuration was changed.

Example

```
$ rladm tune proxy 2 scale_threshold 75
Configuring proxies:
- proxy:2: ok
$ rladm info proxy 2 | grep scale_threshold
scale_threshold: 75 (%)
```

Updated: August 24, 2023

rladmin upgrade

Upgrades the version of a module or Redis Enterprise Software for a database.

upgrade db

Schedules a restart of the primary and replica processes of a database and then upgrades the database to the latest version of Redis Enterprise Software.

For more information, see [Upgrade an existing Redis Software Deployment](#).

```
rladmin upgrade db { db:<id> | <name> }
  [ preserve_roles ]
  [ keep_redis_version ]
  [ discard_data ]
  [ force_discard ]
  [ parallel_shards_upgrade ]
  [ keep_crdt_protocol_version ]
  [ redis_version <version> ]
  [ force ]
  [ { latest_with_modules | and module module_name <module name> version <version> module_args
<arguments string> } ]
```

As of v6.2.4, the default behavior for `upgrade db` has changed. It is now controlled by a new parameter that sets the default upgrade policy used to create new databases and to upgrade ones already in the cluster. To learn more, see [tune cluster default_redis_version](#).

Parameters

Parameters	Type/Value	Description
db	db:<id> name	Database to upgrade
and module	upgrade module command	Clause that allows the upgrade of a database and a specified Redis module in a single step with only one restart (can be specified multiple times)
discard_data		Indicates that data will not be saved after the upgrade

Parameters	Type/Value	Description
force		Forces upgrade and skips warnings and confirmations
force_discard		Forces <code>discard_data</code> if replication or persistence is enabled
keep_crdt_protocol_version		Keeps the current CRDT protocol version
keep_redis_version		Upgrades to a new patch release, not to the latest major.minor version
latest_with_modules		Upgrades the Redis Enterprise Software version and all modules in the database
parallel_shards_upgrade	integer 'all'	Maximum number of shards to upgrade all at once
preserve_roles		Performs an additional failover to guarantee the shards' roles are preserved
redis_version	Redis version	Upgrades the database to the specified version instead of the latest version

Returns

Returns `Done` if the upgrade completed. Otherwise, it returns an error.

Example

```
$ rladm upgrade db db:5
Monitoring e39c8e87-75f9-4891-8c86-78cf151b720b
active - SMUpgradeBDB init
active - SMUpgradeBDB check_slaves
.active - SMUpgradeBDB prepare
active - SMUpgradeBDB stop_forwarding
oactive - SMUpgradeBDB start_wd
active - SMUpgradeBDB wait_for_version
.completed - SMUpgradeBDB
Done
```

upgrade module

Upgrades Redis modules in use by a specific database.

For more information, see [Upgrade modules](#).

```
rladmin upgrade module
  db_name { db:<id> | <name> }
  module_name <mod_name>
  version <version>
  module_args <args_str>
```

Parameters

Parameters	Type/Value	Description
db_name	db:<id> name	Upgrade a module for the specified database
module_name	'ReJSON' 'graph' 'search' 'bf' 'rg' 'timeseries'	Redis module to upgrade
version	module version number	Upgrades the module to the specified version
module_args	'keep_args' string	Module configuration options

For more information about module configuration options, see [Module configuration options](#).

Returns

Returns Done if the upgrade completed. Otherwise, it returns an error.

Example

```
$ rladm upgrade module db_name db:8 module_name graph version 20812 module_args ""  
Monitoring 21ac7659-e44c-4cc9-b243-a07922b2a6cc  
active - SMUpgradeBDB init  
active - SMUpgradeBDB wait_for_version  
0completed - SMUpgradeBDB  
Done
```

Updated: August 17, 2023

rladmin verify

Prints verification reports for the cluster.

verify balance

Prints a balance report that displays all of the unbalanced endpoints or nodes in the cluster.

```
rladmin verify balance [ node <ID> ]
```

The [proxy policy](#) determines which nodes or endpoints to report as unbalanced.

A node is unbalanced if:

- all-nodes proxy policy and the node has no endpoint

An endpoint is unbalanced in the following cases:

- single proxy policy and one of the following is true:
 - Shard placement is [sparse](#) and none of the master shards are on the node
 - Shard placement is [dense](#) and some master shards are on a different node from the endpoint
- all-master-shards proxy policy and one of the following is true:
 - None of the master shards are on the node
 - Some master shards are on a different node from the endpoint

Parameters

Parameter	Type/Value	Description
node	integer	Specify a node ID to return a balance table for that node only (optional)

Returns

Returns a table of unbalanced endpoints and nodes in the cluster.

Examples

Verify all nodes:

```
$ rladm verify balance  
The table presents all of the unbalanced endpoints/nodes in the cluster  
BALANCE:  
NODE:ID DB:ID NAME ENDPOINT:ID PROXY_POLICY LOCAL SHARDS TOTAL SHARDS
```

Verify a specific node:

```
$ rladm verify balance node 1
The table presents all of the unbalanced endpoints/nodes in the cluster
BALANCE:
NODE:ID  DB:ID  NAME  ENDPOINT:ID  PROXY_POLICY  LOCAL SHARDS  TOTAL SHARDS
```

verify rack_aware

Verifies that the cluster complies with the rack awareness policy and reports any discovered rack collisions, if [rack-zone awareness](#) is enabled.

```
rladmin verify rack_aware
```

Parameters

None

Returns

Returns whether the cluster is rack aware. If rack awareness is enabled, it returns any rack collisions.

Example

```
$ rladm verify rack_aware
Cluster policy is not configured for rack awareness.
```

Updated: August 31, 2022

rlcheck

The `rlcheck` utility runs various health checks on a Redis Enterprise Software node and reports any discovered issues. You can use this utility to confirm a successful installation or to verify that the node is functioning properly.

You can run `rlcheck` from the host's command-line interface (CLI). The output of `rlcheck` shows information specific to the host that you run it on.

To open the `rladmin` CLI:

1. Sign in to the Redis Enterprise Software host with an account that is a member of the `redislabs` operating system group.
2. Run: `rlcheck`

The utility runs and reports the result of each check.



Note: To see the `rlcheck` optional flags, run: `rlcheck --help`. Specifically, the `--continue-on-error` flag runs all tests to completion and shows all errors when complete.

To resolve issues reported by `rlcheck`, contact [Redis support](#).

Updated: July 7, 2022

Real-time metrics

The Redis Enterprise Software admin console shows performance metrics for clusters, nodes, databases, and shards.

In the Redis Enterprise admin console, you can see real-time metrics and configure alerts that send notifications based on alert parameters. Select the **Metrics** tab to view the metrics for each component. For more information, see [Monitoring with metrics and alerts](#).

See the following topics for metrics definitions:

- [Database operations](#) for database metrics
- [Resource usage](#) for resource and database usage metrics
- [Auto Tiering](#) for additional metrics for [Auto Tiering](#) databases

Prometheus metrics

To collect and display metrics data from your databases and other cluster components, you can connect your [Prometheus](#) and [Grafana](#) server to your Redis Enterprise Software cluster.

We recommend you use Prometheus and Grafana to view metrics history and trends.

See [Prometheus integration](#) to learn how to connect Prometheus and Grafana to your Redis Enterprise database.

Limitations

Shard limit

Metrics information is not shown for clusters with more than 128 shards. For large clusters, we recommend you use [Prometheus and Grafana](#) to view metrics.

Metrics not shown during shard migration

The following metrics are not measured during [shard migration](#). If you view these metrics while resharding, the graph will be blank.

- [Evicted objects/sec](#)
- [Expired objects/sec](#)
- [Read misses/sec](#)
- [Write misses/sec](#)
- [Total keys](#)
- [Incoming traffic](#)
- [Outgoing traffic](#)
- [Used memory](#)

Updated: August 17, 2023

Auto Tiering Metrics

These metrics are additional metrics for [Auto Tiering](#) databases.

% Values in RAM

Percent of keys whose values are stored in RAM.

Components measured: Database and Shard

Values in flash

Number of keys with values stored in flash, not including [replication](#).

Components measured: Database and Shard

Values in RAM

Number of keys with values stored in RAM, not including [replication](#).

Components measured: Database and Shard

Flash key-value operations

Number of operations on flash key values (read + write + del) per second.

Components measured: Node

Flash bytes/sec

Number of total bytes read and written per second on flash memory.

Components measured: Cluster, Node, Database, and Shard

Flash I/O operations/sec

Number of input/output operations per second on the flash storage device.

Components measured: Cluster and Node

RAM:Flash access ratio

Ratio between logical Redis key value operations and actual flash key value operations.

Components measured: Database and Shard

RAM hit ratio

Ratio of requests processed directly from RAM to total number of requests processed.

Components measured: Database and Shard

Used flash

Total amount of memory used to store values in flash.

Components measured: Database and Shard

Free flash

Amount of free space on flash storage.

Components measured: Cluster and Node

Flash fragmentation

Ratio between the used logical flash memory and the physical flash memory that is used.

Components measured: Database and Shard

Used RAM

Total size of data stored in RAM, including keys, values, overheads, and [replication](#) (if enabled).

Components measured: Database and Shard

RAM dataset overhead

Percentage of the [RAM limit](#) that is used for anything other than values, such as key names, dictionaries, and other overheads.

Components measured: Database and Shard

RAM limit

Maximum amount of RAM that can be used in bytes.

Components measured: Database

RAM usage

Percentage of the [RAM limit](#) used.

Components measured: Database

Storage engine usage

Total count of shards used, filtered by the storage engine (Speedb / RockSB) per given database.

Components measured: Database, Shards

Calculated metrics

These RoF statistics can be calculated from other metrics.

- RoF average key size with overhead

$(\text{ram_dataset_overhead} * \text{used_ram}) / (\text{total_keys} * 2)$

- RoF average value size in RAM

$((1 - \text{ram_dataset_overhead}) * \text{used_ram}) / (\text{values_in_ram} * 2)$

- RoF average value size in flash

$\text{used_flash} / \text{values_in_flash}$

Updated: August 17, 2023

Database operations metrics

Evicted objects/sec

Number of objects evicted from the database per second.

Objects are evicted from the database according to the [eviction policy](#).

Object information is not measured during [shard migration](#).

Components measured: Database and Shard

Expired objects/sec

Number of expired objects per second.

Object information is not measured during [shard migration](#).

Components measured: Database and Shard

Hit ratio

Ratio of the number of operations on existing keys out of the total number of operations.

Components measured: Database and Shard

Read misses/sec

The number of [read operations](#) per second on keys that do not exist.

Read misses are not measured during [shard migration](#).

Components measured: Database

Write misses/sec

Number of [write operations](#) per second on keys that do not exist.

Write misses are not measured during [shard migration](#).

Components measured: Database and Shard

Latency

The total amount of time between sending a Redis operation and receiving a response from the database.

The graph shows average, minimum, maximum, and last latency values for all latency metrics.

Components measured: Database

Reads latency

Latency of read operations.

Components measured: Database

Writes latency

Latency per write operation.

Components measured: Database

Other commands latency

Latency of other operations.

Components measured: Database

Ops/sec

Number of total operations per second, which includes [read operations](#), [write operations](#), and [other operations](#).

Components measured: Cluster, Node, Database, and Shard

Reads/sec

Number of total read operations per second.

To find out which commands are read operations, run the following command with [redis-cli](#):

```
ACL CAT read
```

Components measured: Database

Writes/sec

Number of total write operations per second.

To find out which commands are write operations, run the following command with [redis-cli](#):

```
ACL CAT write
```

Components measured: Database

Pending writes min

Minimum number of write operations queued per [Active-Active](#) replica database.

Pending writes max

Maximum number of write operations queued per [Active-Active](#) replica database.

Other commands/sec

Number of operations per second that are not [read operations](#) or [write operations](#).

Examples of other operations include [PING](#), [AUTH](#), and [INFO](#).

Components measured: Database

Total keys

Total number of keys in the dataset.

Does not include replicated keys, even if [replication](#) is enabled.

Total keys is not measured during [shard migration](#).

Components measured: Database

Resource usage metrics

Connections

Number of connections to the database.

Components measured: Cluster, Node, and Database

CPU usage

Percent of the node CPU used.

Components measured: Cluster and Node

Main thread CPU usage

Percent of the CPU used by the main thread.

Components measured: Database and Shard

Fork CPU usage

CPU usage of Redis child forks.

Components measured: Database and Shard

Total CPU usage

Percent usage of the CPU for all nodes.

Components measured: Database

Free disk space

Remaining unused disk space.

Components measured: Cluster and Node

Memory

Used memory

Total memory used by the database, including RAM, [Flash](#) (if enabled), and [replication](#) (if enabled).

Used memory does not include:

1. Fragmentation overhead - The ratio of memory seen by the operating system to memory allocated by Redis
2. Replication buffers at the primary nodes - Set to 10% of used memory and is between 64 MB and 2048 MB
3. Memory used by Lua scripts - Does not exceed 1 MB
4. Copy on Write (COW) operation that can be triggered by:
 - A full replication process
 - A database snapshot process
 - AOF rewrite process

Used memory is not measured during [shard migration](#).

Components measured: Database and Shard

Free RAM

Available RAM for System use.

Components measured: Cluster and Node

Memory limit

Memory size limit of the database, enforced on the [used memory](#).

Components measured: Database

Memory usage

Percent of memory used by Redis out of the [memory limit](#).

Components measured: Database

Traffic

Incoming traffic

Total incoming traffic to the database in bytes/sec.

All incoming traffic is not measured during [shard migration](#).

Components measured: Cluster, Node and Database

Incoming traffic compressed

Total incoming compressed traffic (in bytes/sec) per [Active-Active](#) replica database.

Incoming traffic uncompressed

Total incoming uncompressed traffic (in bytes/sec) per [Active-Active](#) replica database.

Outgoing traffic

Total outgoing traffic from the database in bytes per second.

Outgoing traffic is not measured during [shard migration](#).

Components measured: Cluster, Node and Database

Updated: August 17, 2023

Redis Enterprise compatibility with open source Redis

Both Redis Enterprise Software and [Redis Enterprise Cloud](#) are compatible with open source Redis (OSS Redis). Redis contributes extensively to the open source Redis project and uses it inside of Redis Enterprise. As a rule, we adhere to the open source project's specifications and update Redis Enterprise with the latest version of open source Redis.

Redis commands

See [Compatibility with open source Redis commands](#) to learn which open source Redis commands are compatible with Redis Enterprise (Software and Cloud).

Configuration settings

[Compatibility with open source Redis configuration settings](#) lists the open source Redis configuration settings supported by Redis Enterprise (Software and Cloud).

Redis clients

You can use any standard [Redis client](#) with Redis Enterprise.

RESP compatibility

Redis Enterprise supports RESP2 and RESP3. See [RESP compatibility with Redis Enterprise](#) for more information.

Compatibility with open source Redis Cluster API

Redis Enterprise supports [Redis OSS Cluster API](#) if it is enabled for a database. For more information, see [Enable OSS Cluster API](#).

Updated: August 17, 2023

Compatibility with open source Redis commands

Learn which open source Redis commands are compatible with Redis Enterprise Software and [Redis Enterprise Cloud](#).

Select a command group for more details about compatibility with standard and Active-Active Redis Enterprise.

Command group	Description
Cluster management	Cluster management commands compatible with Redis Enterprise.
Connection management	Connection management commands compatibility.
Data types	Data type commands compatibility (bitmaps, geospatial indices, hashes, HyperLogLogs, lists, sets, sorted sets, streams, strings).
Keys (generic)	Generic key commands compatible with Redis Enterprise.
Pub/sub	Pub/sub commands compatibility.
Scripting	Scripting and function commands compatibility.
Server management	Server management commands compatibility.
Transactions	Transaction commands compatibility.

Updated: March 1, 2023

Cluster management commands compatibility

Clustering in Redis Enterprise Software and [Redis Enterprise Cloud](#) differs from the [open source Redis cluster](#) and works with all standard Redis clients.

Redis Enterprise blocks most [cluster commands](#). If you try to use a blocked cluster command, it returns an error.

Command	Redis Enterprise	Redis Cloud	Notes
ASKING	Standard Active-Active	Standard Active-Active	
CLUSTER ADDSLOTS	Standard Active-Active	Standard Active-Active	
CLUSTER ADDSLOTSRANGE	Standard Active-Active	Standard Active-Active	
CLUSTER BUMPEPOCH	Standard Active-Active	Standard Active-Active	
CLUSTER COUNT-FAILURE-REPORTS	Standard Active-Active	Standard Active-Active	
CLUSTER COUNTKEYSIN SLOT	Standard Active-Active	Standard Active-Active	
CLUSTER DELSLOTS	Standard Active-Active	Standard Active-Active	
CLUSTER DELSLOTSRANGE	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
CLUSTER FAILOVER	Standard Active-Active	Standard Active-Active	
CLUSTER FLUSHSLOTS	Standard Active-Active	Standard Active-Active	
CLUSTER FORGET	Standard Active-Active	Standard Active-Active	
CLUSTER GETKEYSINSLOT	Standard Active-Active	Standard Active-Active	
CLUSTER HELP	Standard Active-Active	Standard Active-Active	Only supported with the OSS cluster API .
CLUSTER INFO	Standard Active-Active	Standard Active-Active	Only supported with the OSS cluster API .
CLUSTER KEYSLOT	Standard Active-Active	Standard Active-Active	Only supported with the OSS cluster API .
CLUSTER LINKS	Standard Active-Active	Standard Active-Active	
CLUSTER MEET	Standard Active-Active	Standard Active-Active	
CLUSTER MYID	Standard Active-Active	Standard Active-Active	
CLUSTER MYSHARDID	Standard Active-Active	Standard Active-Active	
CLUSTER NODES	Standard Active-Active	Standard Active-Active	Only supported with the OSS cluster API .
CLUSTER REPLICAS	Standard Active-Active	Standard Active-Active	
CLUSTER REPLICATE	Standard Active-Active	Standard Active-Active	
CLUSTER RESET	Standard Active-Active	Standard Active-Active	
CLUSTER SAVECONFIG	Standard Active-Active	Standard Active-Active	
CLUSTER SET-CONFIG-EPOCH	Standard Active-Active	Standard Active-Active	
CLUSTER SETSLOT	Standard Active-Active	Standard Active-Active	
CLUSTER SHARDS	Standard Active-Active	Standard Active-Active	
CLUSTER SLAVES	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v5.0.0.
CLUSTER SLOTS	Standard Active-Active	Standard Active-Active	Only supported with the OSS cluster API . Deprecated as of Redis v7.0.0.
READONLY	Standard Active-Active	Standard Active-Active	
READWRITE	Standard Active-Active	Standard Active-Active	

Updated: August 17, 2023

Connection management commands compatibility

The following tables show which open source Redis [connection management commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Command	Redis Enterprise	Redis Cloud	Notes
AUTH	Standard Active-Active	Standard Active-Active	
CLIENT CACHING	Standard Active-Active	Standard Active-Active	
CLIENT GETNAME	Standard Active-Active	Standard Active-Active	
CLIENT GETREDIR	Standard Active-Active	Standard Active-Active	
CLIENT ID	Standard Active-Active	Standard Active-Active	Because Redis Enterprise clustering allows multiple active proxies , CLIENT ID cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
CLIENT INFO	Standard Active-Active	Standard Active-Active	
CLIENT KILL	Standard Active-Active	Standard Active-Active	
CLIENT LIST	Standard Active-Active	Standard Active-Active	
CLIENT NO-EVICT	Standard Active-Active	Standard Active-Active	
CLIENT NO-TOUCH	Standard Active-Active	Standard Active-Active	
CLIENT PAUSE	Standard Active-Active	Standard Active-Active	
CLIENT REPLY	Standard Active-Active	Standard Active-Active	
CLIENT SETINFO	Standard Active-Active	Standard Active-Active	
CLIENT SETNAME	Standard Active-Active	Standard Active-Active	
CLIENT TRACKING	Standard Active-Active	Standard Active-Active	
CLIENT TRACKINGINFO	Standard Active-Active	Standard Active-Active	
CLIENT UNBLOCK	Standard Active-Active	Standard Active-Active	
CLIENT UNPAUSE	Standard Active-Active	Standard Active-Active	
ECHO	Standard Active-Active	Standard Active-Active	
HELLO	Standard Active-Active	Standard Active-Active	
PING	Standard Active-Active	Standard Active-Active	
QUIT	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v7.2.0.
RESET	Standard Active-Active	Standard Active-Active	
SELECT	Standard Active-Active	Standard Active-Active	Redis Enterprise does not support shared databases due to potential negative performance impacts and blocks any related commands.

Data type commands compatibility

The following tables show which open source Redis data type commands are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Bitmap commands

Command	Redis Enterprise	Redis Cloud	Notes
BITCOUNT	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
BITFIELD	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
BITFIELD_RO	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
BITOP	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
BITPOS	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GETBIT	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
SETBIT	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	

Geospatial indices commands

Command	Redis Enterprise	Redis Cloud	Notes
GEOADD	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GEODIST	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GEOHASH	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GEOPOS	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GEORADIUS	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	Deprecated as of Redis v6.2.0.
GEORADIUS_RO	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	Deprecated as of Redis v6.2.0.
GEORADIUSBYMEMBER	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	Deprecated as of Redis v6.2.0.
GEORADIUSBYMEMBER_RO	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	Deprecated as of Redis v6.2.0.
GEOSEARCH	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	
GEOSEARCHSTORE	⌚ Standard ⌚ Active-Active	⌚ Standard ⌚ Active-Active	

Hash commands

Command	Redis Enterprise	Redis Cloud	Notes
---------	------------------	-------------	-------

Command	Redis Enterprise	Redis Cloud	Notes
HDEL	Standard Active-Active	Standard Active-Active	
HEXISTS	Standard Active-Active	Standard Active-Active	
HGET	Standard Active-Active	Standard Active-Active	
HGETALL	Standard Active-Active	Standard Active-Active	
HINCRBY	Standard Active-Active	Standard Active-Active	
HINCRBYFLOAT	Standard Active-Active	Standard Active-Active	
HKEYS	Standard Active-Active	Standard Active-Active	
HLEN	Standard Active-Active	Standard Active-Active	
HMGET	Standard Active-Active	Standard Active-Active	
HMSET	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v4.0.0.
HRANDFIELD	Standard Active-Active	Standard Active-Active	
HSCAN	Standard Active-Active	Standard Active-Active	
HSET	Standard Active-Active	Standard Active-Active	
HSETNX	Standard Active-Active	Standard Active-Active	
HSTRLEN	Standard Active-Active	Standard Active-Active	
HVALS	Standard Active-Active	Standard Active-Active	

HyperLogLog commands

Command	Redis Enterprise	Redis Cloud	Notes
PFADD	Standard Active-Active	Standard Active-Active	
PFCOUNT	Standard Active-Active	Standard Active-Active	
PFDEBUG	Standard Active-Active	Standard Active-Active	
PFMERGE	Standard Active-Active	Standard Active-Active	
PFSELFTEST	Standard Active-Active	Standard Active-Active	

List commands

Command	Redis Enterprise	Redis Cloud	Notes
BLMOVE	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
BLMPOP	Standard Active-Active	Standard Active-Active	
BLPOP	Standard Active-Active	Standard Active-Active	
BRPOP	Standard Active-Active	Standard Active-Active	
BRPOPLPUSH	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
LINDEX	Standard Active-Active	Standard Active-Active	
LINSERT	Standard Active-Active	Standard Active-Active	
LLEN	Standard Active-Active	Standard Active-Active	
LMOVE	Standard Active-Active	Standard Active-Active	
LMPOP	Standard Active-Active	Standard Active-Active	
LPOP	Standard Active-Active	Standard Active-Active	
LPOS	Standard Active-Active	Standard Active-Active	
LPUSH	Standard Active-Active	Standard Active-Active	
LPUSHX	Standard Active-Active	Standard Active-Active	
LRANGE	Standard Active-Active	Standard Active-Active	
LREM	Standard Active-Active	Standard Active-Active	
LSET	Standard Active-Active	Standard Active-Active	
LTRIM	Standard Active-Active	Standard Active-Active	
RPOP	Standard Active-Active	Standard Active-Active	
RPOPLPUSH	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
RPUSH	Standard Active-Active	Standard Active-Active	
RPUSHX	Standard Active-Active	Standard Active-Active	

Set commands

Command	Redis Enterprise	Redis Cloud	Notes
SADD	Standard Active-Active	Standard Active-Active	
SCARD	Standard Active-Active	Standard Active-Active	
SDIFF	Standard Active-Active	Standard Active-Active	
SDIFFSTORE	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
SINTER	Standard Active-Active	Standard Active-Active	
SINTERCARD	Standard Active-Active	Standard Active-Active	
SINTERSTORE	Standard Active-Active	Standard Active-Active	
SISMEMBER	Standard Active-Active	Standard Active-Active	
SMEMBERS	Standard Active-Active	Standard Active-Active	
SMISMEMBER	Standard Active-Active	Standard Active-Active	
SMOVE	Standard Active-Active	Standard Active-Active	
SPOP	Standard Active-Active	Standard Active-Active	
SRANDMEMBER	Standard Active-Active	Standard Active-Active	
SREM	Standard Active-Active	Standard Active-Active	
SSCAN	Standard Active-Active	Standard Active-Active	
SUNION	Standard Active-Active	Standard Active-Active	
SUNIONSTORE	Standard Active-Active	Standard Active-Active	

Sorted set commands

Command	Redis Enterprise	Redis Cloud	Notes
BZMPOP	Standard Active-Active	Standard Active-Active	
BZPOPMAX	Standard Active-Active	Standard Active-Active	
BZPOPMIN	Standard Active-Active	Standard Active-Active	
ZADD	Standard Active-Active	Standard Active-Active	
ZCARD	Standard Active-Active	Standard Active-Active	
ZCOUNT	Standard Active-Active	Standard Active-Active	
ZDIFF	Standard Active-Active	Standard Active-Active	
ZDIFFSTORE	Standard Active-Active	Standard Active-Active	
ZINCRBY	Standard Active-Active	Standard Active-Active	
ZINTER	Standard Active-Active	Standard Active-Active	
ZINTERCARD	Standard Active-Active	Standard Active-Active	
ZINTERSTORE	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
ZLEXCOUNT	Standard Active-Active	Standard Active-Active	
ZMPOP	Standard Active-Active	Standard Active-Active	
ZMSCORE	Standard Active-Active	Standard Active-Active	
ZPOPMAX	Standard Active-Active	Standard Active-Active	
ZPOPMIN	Standard Active-Active	Standard Active-Active	
ZRANDOMMEMBER	Standard Active-Active	Standard Active-Active	
ZRANGE	Standard Active-Active	Standard Active-Active	
ZRANGEBYLEX	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
ZRANGEBYSCORE	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
ZRANGESTORE	Standard Active-Active	Standard Active-Active	
ZRANK	Standard Active-Active	Standard Active-Active	
ZREM	Standard Active-Active	Standard Active-Active	
ZREMRANGEBYLEX	Standard Active-Active	Standard Active-Active	
ZREMRANGEBYRANK	Standard Active-Active	Standard Active-Active	
ZREMRANGEBYSCORE	Standard Active-Active	Standard Active-Active	
ZRVRANGE	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
ZREVRANGEBYLEX	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
ZREVRANGEBYSCORE	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
ZREVRANK	Standard Active-Active	Standard Active-Active	
ZSCAN	Standard Active-Active	Standard Active-Active	
ZSCORE	Standard Active-Active	Standard Active-Active	
ZUNION	Standard Active-Active	Standard Active-Active	
ZUNIONSTORE	Standard Active-Active	Standard Active-Active	

Stream commands

Command	Redis Enterprise	Redis Cloud	Notes
XACK	Standard Active-Active	Standard Active-Active	
XADD	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
XAUTOCLAIM	Standard Active-Active	Standard Active-Active	
XCLAIM	Standard Active-Active	Standard Active-Active	
XDEL	Standard Active-Active	Standard Active-Active	
XGROUP	Standard Active-Active	Standard Active-Active	
XINFO	Standard Active-Active	Standard Active-Active	
XLEN	Standard Active-Active	Standard Active-Active	
XPENDING	Standard Active-Active	Standard Active-Active	
XRANGE	Standard Active-Active	Standard Active-Active	
XREAD	Standard Active-Active	Standard Active-Active	
XREADGROUP	Standard Active-Active	Standard Active-Active	
XREVRANGE	Standard Active-Active	Standard Active-Active	
XSETID	Standard Active-Active	Standard Active-Active	
XTRIM	Standard Active-Active	Standard Active-Active	

String commands

Command	Redis Enterprise	Redis Cloud	Notes
APPEND	Standard Active-Active	Standard Active-Active	
DECR	Standard Active-Active	Standard Active-Active	
DECRBY	Standard Active-Active	Standard Active-Active	
GET	Standard Active-Active	Standard Active-Active	
GETDEL	Standard Active-Active	Standard Active-Active	
GETEX	Standard Active-Active*	Standard Active-Active*	*Not supported for HyperLogLog.
GETRANGE	Standard Active-Active	Standard Active-Active	
GETSET	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v6.2.0.
INCR	Standard Active-Active	Standard Active-Active	
INCRBY	Standard Active-Active	Standard Active-Active	
INCRBYFLOAT	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
LCS	Standard Active-Active	Standard Active-Active	
MGET	Standard Active-Active	Standard Active-Active	
MSET	Standard Active-Active	Standard Active-Active	
MSETNX	Standard Active-Active	Standard Active-Active	
PSETEX	Standard Active-Active	Standard Active-Active	
SET	Standard Active-Active	Standard Active-Active	
SETEX	Standard Active-Active	Standard Active-Active	
SETNX	Standard Active-Active	Standard Active-Active	
SETRANGE	Standard Active-Active	Standard Active-Active	
STRALGO	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v7.0.0.
STRLEN	Standard Active-Active	Standard Active-Active	
SUBSTR	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v2.0.0.

Updated: August 17, 2023

Key commands compatibility

The following table shows which open source Redis [key \(generic\) commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Command	Redis Enterprise	Redis Cloud	Notes
COPY	Standard Active-Active*	Standard Active-Active*	For Active-Active or clustered databases, the source and destination keys must be in the same hash slot. *Not supported for stream consumer group info.
DEL	Standard Active-Active	Standard Active-Active	
DUMP	Standard Active-Active	Standard Active-Active	
EXISTS	Standard Active-Active	Standard Active-Active	
EXPIRE	Standard Active-Active	Standard Active-Active	
EXPIREAT	Standard Active-Active	Standard Active-Active	
EXPIRETIME	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
KEYS	Standard Active-Active	Standard Active-Active	
MIGRATE	Standard Active-Active	Standard Active-Active	
MOVE	Standard Active-Active	Standard Active-Active	Redis Enterprise does not support shared databases due to potential negative performance impacts and blocks any related commands.
OBJECT ENCODING	Standard Active-Active	Standard Active-Active	
OBJECT FREQ	Standard Active-Active	Standard Active-Active	
OBJECT IDLETIME	Standard Active-Active	Standard Active-Active	
OBJECT REFCOUNT	Standard Active-Active	Standard Active-Active	
PERSIST	Standard Active-Active	Standard Active-Active	
PEXPIRE	Standard Active-Active	Standard Active-Active	
PEXPIREAT	Standard Active-Active	Standard Active-Active	
PEXPIRETIME	Standard Active-Active	Standard Active-Active	
PTTL	Standard Active-Active	Standard Active-Active	
RANDOMKEY	Standard Active-Active	Standard Active-Active	
RENAME	Standard Active-Active*	Standard Active-Active*	For Active-Active or clustered databases, the original key and new key must be in the same hash slot. *Not supported for stream consumer group info.
RENAMENX	Standard Active-Active	Standard Active-Active	For Active-Active or clustered databases, the original key and new key must be in the same hash slot.
RESTORE	Standard Active-Active*	Standard Active-Active*	*Only supported for module keys.
SCAN	Standard Active-Active	Standard Active-Active	
SORT	Standard Active-Active	Standard Active-Active	
SORT_RO	Standard Active-Active	Standard Active-Active	
TOUCH	Standard Active-Active	Standard Active-Active	
TTL	Standard Active-Active	Standard Active-Active	
TYPE	Standard Active-Active	Standard Active-Active	
UNLINK	Standard Active-Active	Standard Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
WAIT	Standard Active-Active*	Standard Active-Active	*For Active-Active databases, WAIT commands are supported for primary and replica shard replication. You can contact support to enable WAIT for local replicas only. WAIT is not supported for cross-instance replication.
WAITAOF	Standard Active-Active	Standard Active-Active	

Updated: August 17, 2023

Pub/sub commands compatibility

The following table shows which open source Redis [pub/sub commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Command	Redis Enterprise	Redis Cloud	Notes
PSUBSCRIBE	Standard Active-Active	Standard Active-Active	
PUBLISH	Standard Active-Active	Standard Active-Active	
PUBSUB CHANNELS	Standard Active-Active	Standard Active-Active	
PUBSUB NUMPAT	Standard Active-Active	Standard Active-Active	
PUBSUB NUMSUB	Standard Active-Active	Standard Active-Active	
PUBSUB SHARDCHANNELS	Standard Active-Active	Standard Active-Active	
PUBSUB SHARDNUMSUB	Standard Active-Active	Standard Active-Active	
PUNSUBSCRIBE	Standard Active-Active	Standard Active-Active	
SPUBLISH	Standard Active-Active	Standard Active-Active	
SSUBSCRIBE	Standard Active-Active	Standard Active-Active	
SUBSCRIBE	Standard Active-Active	Standard Active-Active	
SUNSUBSCRIBE	Standard Active-Active	Standard Active-Active	
UNSUBSCRIBE	Standard Active-Active	Standard Active-Active	

Updated: August 17, 2023

Scripting commands compatibility

The following table shows which open source Redis [scripting and function commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Function commands

Command	Redis Enterprise	Redis Cloud	Notes
FCALL	Standard Active-Active	Standard Active-Active	
FCALL_RO	Standard Active-Active	Standard Active-Active	
FUNCTION DELETE	Standard Active-Active	Standard Active-Active	
FUNCTION DUMP	Standard Active-Active	Standard Active-Active	
FUNCTION FLUSH	Standard Active-Active	Standard Active-Active	
FUNCTION HELP	Standard Active-Active	Standard Active-Active	
FUNCTION KILL	Standard Active-Active	Standard Active-Active	
FUNCTION LIST	Standard Active-Active	Standard Active-Active	
FUNCTION LOAD	Standard Active-Active	Standard Active-Active	
FUNCTION RESTORE	Standard Active-Active	Standard Active-Active	
FUNCTION STATS	Standard Active-Active	Standard Active-Active	

Scripting commands

Command	Redis Enterprise	Redis Cloud	Notes
EVAL	Standard Active-Active	Standard Active-Active	
EVAL_RO	Standard Active-Active	Standard Active-Active	
EVALSHA	Standard Active-Active	Standard Active-Active	
EVALSHA_RO	Standard Active-Active	Standard Active-Active	
SCRIPT DEBUG	Standard Active-Active	Standard Active-Active	
SCRIPT EXISTS	Standard Active-Active	Standard Active-Active	
SCRIPT FLUSH	Standard Active-Active	Standard Active-Active	
SCRIPT KILL	Standard Active-Active	Standard Active-Active	
SCRIPT LOAD	Standard Active-Active	Standard Active-Active	

Updated: August 17, 2023

Server management commands compatibility

The following tables show which open source Redis [server management commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Access control commands

Several access control list (ACL) commands are not available in Redis Enterprise. Instead, you can manage access controls from the admin consoles for [Redis Enterprise Software](#) and [Redis Cloud](#).

Command	Redis Enterprise	Redis Cloud	Notes
ACL CAT	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .
ACL DELUSER	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACL DRYRUN	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Might reply with “unknown user” for LDAP users even if AUTH succeeds.
ACL GENPASS	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACL GETUSER	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .
ACL HELP	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .
ACL LIST	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .
ACL LOAD	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACL LOG	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACLSAVE	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACLSETUSER	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
ACL USERS	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .
ACL WHOAMI	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Not supported for scripts .

Configuration commands

Command	Redis Enterprise	Redis Cloud	Notes
CONFIG GET	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Only supports a subset of configuration settings.
CONFIG RESETSTAT	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
CONFIG REWRITE	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	
CONFIG SET	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	Only supports a subset of configuration settings.

General server commands

Command	Redis Enterprise	Redis Cloud	Notes
COMMAND	⌘ Standard ⌘ Active-Active	⌘ Standard ⌘ Active-Active	

Command	Redis Enterprise	Redis Cloud	Notes
COMMAND COUNT	Standard Active-Active	Standard Active-Active	
COMMAND DOCS	Standard Active-Active	Standard Active-Active	
COMMAND GETKEYS	Standard Active-Active	Standard Active-Active	
COMMAND GETKEYSANDFLAGS	Standard Active-Active	Standard Active-Active	
COMMAND HELP	Standard Active-Active	Standard Active-Active	
COMMAND INFO	Standard Active-Active	Standard Active-Active	
COMMAND LIST	Standard Active-Active	Standard Active-Active	
DEBUG	Standard Active-Active	Standard Active-Active	
FLUSHALL	Standard Active-Active*	Standard Active-Active	*Can use the Active-Active flush API request .
FLUSHDB	Standard Active-Active*	Standard Active-Active	*Can use the Active-Active flush API request .
LOLWUT	Standard Active-Active	Standard Active-Active	
SHUTDOWN	Standard Active-Active	Standard Active-Active	
SWAPDB	Standard Active-Active	Standard Active-Active	
TIME	Standard Active-Active	Standard Active-Active	

Module commands

For Redis Enterprise Software, you can [manage Redis modules](#) from the admin console or with [REST API requests](#).

Redis Cloud manages modules for you and lets you [enable modules](#) when you create a database.

Command	Redis Enterprise	Redis Cloud	Notes
MODULE HELP	Standard Active-Active	Standard Active-Active	
MODULE LIST	Standard Active-Active	Standard Active-Active	
MODULE LOAD	Standard Active-Active	Standard Active-Active	
MODULE LOADEX	Standard Active-Active	Standard Active-Active	
MODULE UNLOAD	Standard Active-Active	Standard Active-Active	

Monitoring commands

Although Redis Enterprise does not support certain monitoring commands, you can use the admin consoles to view Redis Enterprise Software [metrics](#) and [logs](#) or Redis Cloud [metrics](#) and [logs](#).

Command	Redis Enterprise	Redis Cloud	Notes

Command	Redis Enterprise	Redis Cloud	Notes
DBSIZE	Standard Active-Active	Standard Active-Active	
INFO	Standard Active-Active	Standard Active-Active	Not supported for scripts .
LATENCY DOCTOR	Standard Active-Active	Standard Active-Active	
LATENCY GRAPH	Standard Active-Active	Standard Active-Active	
LATENCY HELP	Standard Active-Active	Standard Active-Active	
LATENCY HISTOGRAM	Standard Active-Active	Standard Active-Active	
LATENCY HISTORY	Standard Active-Active	Standard Active-Active	
LATENCY LATEST	Standard Active-Active	Standard Active-Active	
LATENCY RESET	Standard Active-Active	Standard Active-Active	
MEMORY DOCTOR	Standard Active-Active	Standard Active-Active	
MEMORY HELP	Standard Active-Active	Standard Active-Active	
MEMORY MALLOC-STATS	Standard Active-Active	Standard Active-Active	
MEMORY PURGE	Standard Active-Active	Standard Active-Active	
MEMORY STATS	Standard Active-Active	Standard Active-Active	
MEMORY USAGE	Standard Active-Active	Standard Active-Active	
MONITOR	Standard Active-Active	Standard Active-Active	
SLOWLOG GET	Standard Active-Active	Standard Active-Active	Not supported for scripts .
SLOWLOG LEN	Standard Active-Active	Standard Active-Active	Not supported for scripts .
SLOWLOG RESET	Standard Active-Active	Standard Active-Active	Not supported for scripts .

Persistence commands

Data persistence and backup commands are not available in Redis Enterprise. Instead, you can [manage data persistence](#) and [backups](#) from the admin consoles for Redis Enterprise Software and [Redis Cloud](#).

Command	Redis Enterprise	Redis Cloud	Notes
BGREWRITEAOF	Standard Active-Active	Standard Active-Active	
BGSAVE	Standard Active-Active	Standard Active-Active	
LASTSAVE	Standard Active-Active	Standard Active-Active	
SAVE	Standard Active-Active	Standard Active-Active	

Replication commands

Redis Enterprise automatically manages replication.

Command	Redis Enterprise	Redis Cloud	Notes
FAILOVER	Standard Active-Active	Standard Active-Active	
MIGRATE	Standard Active-Active	Standard Active-Active	
PSYNC	Standard Active-Active	Standard Active-Active	
REPLCONF	Standard Active-Active	Standard Active-Active	
REPLICAOF	Standard Active-Active	Standard Active-Active	
RESTORE-ASKING	Standard Active-Active	Standard Active-Active	
ROLE	Standard Active-Active	Standard Active-Active	
SLAVEOF	Standard Active-Active	Standard Active-Active	Deprecated as of Redis v5.0.0.
SYNC	Standard Active-Active	Standard Active-Active	

Updated: August 17, 2023

Transaction commands compatibility

The following table shows which open source Redis [transaction commands](#) are compatible with standard and Active-Active databases in Redis Enterprise Software and Redis Enterprise Cloud.

Command	Redis Enterprise	Redis Cloud	Notes
DISCARD	Standard Active-Active	Standard Active-Active	
EXEC	Standard Active-Active	Standard Active-Active	
MULTI	Standard Active-Active	Standard Active-Active	
UNWATCH	Standard Active-Active	Standard Active-Active	
WATCH	Standard Active-Active	Standard Active-Active	

Updated: December 7, 2022

Compatibility with open source Redis configuration settings

Redis Enterprise Software and Redis Enterprise Cloud only support a subset of [open source Redis configuration settings](#). Using `CONFIG GET` or `CONFIG SET` with unsupported configuration settings returns an error.

Setting	Redis Enterprise	Redis Cloud	Notes
activererehashing	Standard Active-Active	Standard Active-Active	
busy-reply-threshold	Standard Active-Active	Standard Active-Active	Value must be between 0 and 60000.
hash-max-listpack-entries	Standard Active-Active	Standard Active-Active	
hash-max-listpack-value	Standard Active-Active	Standard Active-Active	
hash-max-ziplist-entries	Standard Active-Active	Standard Active-Active	
hash-max-ziplist-value	Standard Active-Active	Standard Active-Active	
hll-sparse-max-bytes	Standard Active-Active	Standard Active-Active	
list-compress-depth	Standard Active-Active	Standard Active-Active	
list-max-listpack-size	Standard Active-Active	Standard Active-Active	
list-max-ziplist-size	Standard Active-Active	Standard Active-Active	
lua-time-limit	Standard Active-Active	Standard Active-Active	Value must be between 0 and 60000.
notify-keyspace-events	Standard Active-Active	Standard Active-Active	
set-max-intset-entries	Standard Active-Active	Standard Active-Active	
slowlog-log-slower-than	Standard Active-Active	Standard Active-Active	Value must be larger than 1000.
slowlog-max-len	Standard Active-Active	Standard Active-Active	Value must be between 128 and 1024.
stream-node-max-bytes	Standard Active-Active	Standard Active-Active	
stream-node-max-entries	Standard Active-Active	Standard Active-Active	
zset-max-listpack-entries	Standard Active-Active	Standard Active-Active	
zset-max-listpack-value	Standard Active-Active	Standard Active-Active	
zset-max-ziplist-entries	Standard Active-Active	Standard Active-Active	
zset-max-ziplist-value	Standard Active-Active	Standard Active-Active	

Updated: March 1, 2023

RESP compatibility with Redis Enterprise

RESP (Redis Serialization Protocol) is the protocol that clients use to communicate with Redis databases. See the [RESP protocol specification](#) for more information.

Supported RESP versions

- RESP2 is supported by all Redis Enterprise versions.

- RESP3 is supported by Redis Enterprise 7.2 and later.

Enable RESP3 for a database

To use RESP3 with a Redis Enterprise Software database:

1. Upgrade Redis servers to version 7.2 or later.

For Active-Active and Replica Of databases:

1. Upgrade all participating clusters to Redis Enterprise version 7.2.x or later.
2. Upgrade all databases to version 7.x or later.

2. Enable RESP3 support for your database (enabled by default):

- `rladmin tune db`:

```
rladmin tune db db:<ID> resp3 enabled
```

- [Update database configuration](#) REST API request:

```
PUT /v1/b dbs/<database_id>
{ "resp3": true }
```

Deactivate RESP3 for a database

To deactivate RESP3 support for a database:

- `rladmin tune db`:

```
rladmin tune db db:<ID> resp3 disabled
```

- [Update database configuration](#) REST API request:

```
PUT /v1/b dbs/<database_id>
{ "resp3": false }
```

When RESP3 is deactivated, connected clients that use RESP3 are disconnected from the database.



Note: You cannot use sharded pub/sub if you deactivate RESP3 support.

Change default RESP3 option

The cluster-wide option `resp3_default` determines the default value of the `resp3` option, which enables or deactivates RESP3 for a database, upon upgrading a database to version 7.2. `resp3_default` is set to `enabled` by default.

To change `resp3_default` to `disabled`, use one of the following methods:

- `rladmin tune cluster`

```
rladmin tune cluster resp3_default disabled
```

- [Update cluster policy](#) REST API request:

```
PUT /v1/cluster/policy
{ "resp3_default": false }
```

Client prerequisites for Redis 7.2 upgrade

The Redis clients [Go-Redis](#) version 9 and [Lettuce](#) versions 6 and later use RESP3 by default. If you use either client to run Redis Stack commands, you should set the client's protocol version to RESP2 before upgrading your database to Redis version 7.2 to prevent potential application issues due to RESP3 breaking.

changes.

Go-Redis

For applications using Go-Redis v9.0.5 or later, set the protocol version to RESP2:

```
client := redis.NewClient(&redis.Options{
    Addr:     "<database_endpoint>",
    Protocol: 2, // Pin the protocol version
})
```

Lettuce

To set the protocol version to RESP2 with Lettuce v6 or later:

```
import io.lettuce.core.*;
import io.lettuce.core.api.*;
import io.lettuce.core.protocol.ProtocolVersion;

// ...
RedisClient client = RedisClient.create("<database_endpoint>");
client.setOptions(ClientOptions.builder()
    .protocolVersion(ProtocolVersion.RESP2) // Pin the protocol version
    .build());
// ...
```

If you are using [LettuceMod](#), you need to upgrade to [v3.6.0](#).

Updated: August 17, 2023

REST API

Redis Enterprise Software provides a REST API to help you automate common tasks.

Here, you'll find the details of the API and how to use it.

For more info, see:

- Supported [request endpoints](#), organized by path
- Supported [objects](#), both request and response
- Built-in roles and associated [permissions](#)
- [Redis Enterprise Software REST API quick start](#) with examples

Authentication

Authentication to the Redis Enterprise Software API occurs via [Basic Auth](#). Provide your username and password as the basic auth credentials.

If the username and password is incorrect or missing, the request will fail with a [401 Unauthorized](#) status code.

Example request using [cURL](#):

```
curl -u "demo@redislabs.com:password" \
      https://localhost:9443/v1/b dbs
```

For more examples, see the [Redis Enterprise Software REST API quick start](#)

Permissions

By default, the admin user is authorized for access to all endpoints. Use [role-based access controls](#) and [role permissions](#) to manage access.

If a user attempts to access an endpoint that is not allowed in their role, the request will fail with a [403 Forbidden](#) status code. For more details on which user roles can access certain endpoints, see [Permissions](#).

Certificates

The Redis Enterprise Software REST API uses [Self-signed certificates](#) to ensure the product is secure. When you use the default self-signed certificates, the HTTPS requests will fail with SSL certificate problem: self signed certificate unless you turn off SSL certificate verification. The examples in this tutorial turn off SSL certificate verification.

Ports

All calls must be made over SSL to port 9443. For the API to work, port 9443 must be exposed to incoming traffic or mapped to a different port.

If you are using a [Redis Enterprise Software Docker image](#), run the following command to start the Docker image with port 9443 exposed:

```
docker run -p 9443:9443 redislabs/redis
```

Versions

All API requests are versioned in order to minimize the impact of backwards-incompatible API changes and to coordinate between different versions operating in parallel.

Specify the version in the request [URI](#), as shown in the following table:

Request path	Description
POST /v1/bdbs	A version 1 request for the /bdbs endpoint.
POST /v2/bdbs	A version 2 request for the /bdbs endpoint.

When an endpoint supports multiple versions, each version is documented on the corresponding endpoint. For example, the [bdbs request page](#) documents POST requests for [version 1](#) and [version 2](#).

Headers

Requests

Redis Enterprise REST API requests support the following HTTP headers:

Header	Supported/Required Values
Accept	application/json
Content-Length	Length (in bytes) of request message
Content-Type	application/json (required for PUT or POST requests)

If the client specifies an invalid header, the request will fail with a [400 Bad Request](#) status code.

Responses

Redis Enterprise REST API responses support the following HTTP headers:

Header	Supported/Required Values
Content-Type	application/json
Content-Length	Length (in bytes) of response message

JSON requests and responses

The Redis Enterprise Software REST API uses [JavaScript Object Notation \(JSON\)](#) for requests and responses. See the [RFC 4627 technical specifications](#) for additional information about JSON.

Some responses may have an empty body but indicate the response with standard [HTTP codes](#).

Both requests and responses may include zero or more objects.

If the request is for a single entity, the response returns a single JSON object or none. If the request is for a list of entities, the response returns a JSON array with zero or more elements.

If you omit certain JSON object fields from a request, they may be assigned default values, which often indicate that these fields are not in use.

Response types and error codes

[HTTP status codes](#) indicate the result of an API request. This can be 200 OK if the server accepted the request, or it can be one of many error codes.

The most common responses for a Redis Enterprise API request are:

Response	Condition/Required handling
200 OK	Success
400 Bad Request	The request failed, generally due to a typo or other mistake.
401 Unauthorized	The request failed because the authentication information was missing or incorrect.
403 Forbidden	The user cannot access the specified URI .
404 Not Found	The URI does not exist.
503 Service Unavailable	The node is not responding or is not a member of the cluster.
505 HTTP Version Not Supported	An unsupported x-api-version was used. See versions .

Some endpoints return different response codes. The request references for these endpoints document these special cases.

Updated: August 17, 2023

Redis Enterprise Software REST API quick start

Redis Enterprise Software includes a REST API that allows you to automate certain tasks. This article shows you how to send a request to the Redis Enterprise Software REST API.

Fundamentals

No matter which method you use to send API requests, there are a few common concepts to remember.

Type	Description
Authentication	Use Basic Auth with your cluster username (email) and password
Ports	All calls are made to port 9443 by default
Versions	Specify the version in the request URI
Headers	Accept and Content-Type should be application/json
Response types and error codes	A response of 200 OK means success; otherwise, the request failed due to an error

For more information, see [Redis Enterprise Software REST API](#).

cURL example requests

[cURL](#) is a command-line tool that allows you to send HTTP requests from a terminal.

You can use the following options to build a cURL request:

Option	Description
-X	Method (GET, PUT, PATCH, POST, or DELETE)
-H	Request header, can be specified multiple times
-u	Username and password information
-d	JSON data for PUT or POST requests

Option	Description
-F	Form data for PUT or POST requests, such as for the POST /v1/modules or POST /v2/modules endpoint
-k	Turn off SSL verification
-i	Show headers and status code as well as the response body

See the [cURL documentation](#) for more information.

GET request

Use the following cURL command to get a list of databases with the [GET /v1/b dbs](#) endpoint.

```
$ curl -X GET -H "accept: application/json" \
      -u "[username]:[password]" \
      https://[host][:port]/v1/b dbs -k -i

HTTP/1.1 200 OK
server: envoy
date: Tue, 14 Jun 2022 19:24:30 GMT
content-type: application/json
content-length: 2833
cluster-state-id: 42
x-envoy-upstream-service-time: 25

[
  {
    ...
    "name": "tr01",
    ...
    "uid": 1,
    "version": "6.0.16",
    "wait_command": true
  }
]
```

In the response body, the uid is the database ID. You can use the database ID to view or update the database using the API.

For more information about the fields returned by [GET /v1/b dbs](#), see the [b dbs object](#).

PUT request

Once you have the database ID, you can use [PUT /v1/b dbs](#) to update the configuration of the database.

For example, you can pass the database uid 1 as a URL parameter and use the -d option to specify the new name when you send the request. This changes the database's name from tr01 to database1:

```
$ curl -X PUT -H "accept: application/json" \
      -H "content-type: application/json" \
      -u "cameron.bates@redis.com:test123" \
      https://[host]:[port]/v1/b dbs/1 \
      -d '{ "name": "database1" }' -k -i
HTTP/1.1 200 OK
server: envoy
date: Tue, 14 Jun 2022 20:00:25 GMT
content-type: application/json
content-length: 2933
cluster-state-id: 43
x-envoy-upstream-service-time: 159

{
  ...
  "name" : "database1",
  ...
  "uid" : 1,
  "version" : "6.0.16",
  "wait_command" : true
}
```

For more information about the fields you can update with [PUT /v1/b dbs/](#), see the [b dbs object](#).

Client examples

You can also use client libraries to make API requests in your preferred language.

To follow these examples, you need:

- A [Redis Enterprise Software](#) node
- Python 3 and the [requests](#) Python library
- [node.js](#) and [node-fetch](#)

Python

```

import json
import requests

# Required connection information - replace with your host, port, username, and password
host = "[host]"
port = "[port]"
username = "[username]"
password = "[password]"

# Get the list of databases using GET /v1/b dbs
b dbs_uri = "https://{}:{}/v1/b dbs".format(host, port)

print("GET {}".format(b dbs_uri))
get_resp = requests.get(b dbs_uri,
    auth = (username, password),
    headers = { "accept" : "application/json" },
    verify = False)

print("{} {}".format(get_resp.status_code, get_resp.reason))
for header in get_resp.headers.keys():
    print("{}: {}".format(header, get_resp.headers[header]))

print("\n" + json.dumps(get_resp.json(), indent=4))

# Rename all databases using PUT /v1/b dbs
for b db in get_resp.json():
    uid = b db["uid"] # Get the database ID from the JSON response

    put_uri = "{}/{}".format(b dbs_uri, uid)
    new_name = "database{}".format(uid)
    put_data = { "name" : new_name }

    print("PUT {} {}".format(put_uri, json.dumps(put_data)))

    put_resp = requests.put(put_uri,
        data = json.dumps(put_data),
        auth = (username, password),
        headers = { "content-type" : "application/json" },
        verify = False)

    print("{} {}".format(put_resp.status_code, put_resp.reason))
    for header in put_resp.headers.keys():
        print("{}: {}".format(header, put_resp.headers[header]))

    print("\n" + json.dumps(put_resp.json(), indent=4))

```

See the [Python requests library documentation](#) for more information.

Output

```

$ python rs_api.py
python rs_api.py
GET https://[host]:[port]/v1/b dbs
InsecureRequestWarning: Unverified HTTPS request is being made to host '[host]'.
Adding certificate verification is strongly advised.
See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
    warnings.warn(
200 OK
server: envoy
date: Wed, 15 Jun 2022 15:49:43 GMT
content-type: application/json
content-length: 2832
cluster-state-id: 89
x-envoy-upstream-service-time: 27

[
    {
        ...
        "name": "tr01",
        ...
        "uid": 1,
        "version": "6.0.16",
        "wait_command": true
    }
]

PUT https://[host]:[port]/v1/b dbs/1 {"name": "database1"}
InsecureRequestWarning: Unverified HTTPS request is being made to host '[host]'.
Adding certificate verification is strongly advised.
See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
    warnings.warn(
200 OK
server: envoy
date: Wed, 15 Jun 2022 15:49:43 GMT
content-type: application/json
content-length: 2933
cluster-state-id: 90
x-envoy-upstream-service-time: 128

{
    ...
    "name" : "database1",
    ...
    "uid" : 1,
    "version" : "6.0.16",
    "wait_command" : true
}

```

node.js

```

import fetch, { Headers } from 'node-fetch';
import * as https from 'https';

const HOST = '[host]';
const PORT = '[port]';
const USERNAME = '[username]';
const PASSWORD = '[password]';

// Get the list of databases using GET /v1/b dbs
const BDBS_URI = `https://${HOST}:${PORT}/v1/b dbs`;
const USER_CREDENTIALS = Buffer.from(` ${USERNAME} : ${PASSWORD}`).toString('base64');
const AUTH_HEADER = `Basic ${USER_CREDENTIALS}`;

console.log(`GET ${BDBS_URI}`);

const HTTPS_AGENT = new https.Agent({
    rejectUnauthorized: false
});

const response = await fetch(BDBS_URI, {
    method: 'GET',
    headers: {
        'Accept': 'application/json',
        'Authorization': AUTH_HEADER
    },
    agent: HTTPS_AGENT
});

const responseObject = await response.json();
console.log(` ${response.status}: ${response.statusText}`);
console.log(responseObject);

// Rename all databases using PUT /v1/b dbs
for (const database of responseObject) {
    const DATABASE_URI = `${BDBS_URI}/${database.uid}`;
    const new_name = `database${database.uid}`;

    console.log(`PUT ${DATABASE_URI}`);

    const response = await fetch(DATABASE_URI, {
        method: 'PUT',
        headers: {
            'Authorization': AUTH_HEADER,
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            'name': new_name
        }),
        agent: HTTPS_AGENT
    });

    console.log(` ${response.status}: ${response.statusText}`);
    console.log(await(response.json()));
}

```

See the [node-fetch documentation](#) for more info.

Output

```

$ node rs_api.js
GET https://[host]:[port]/v1/b dbs
200: OK
[
  {
    ...
    "name": "tr01",
    ...
    "slave_ha" : false,
    ...
    "uid": 1,
    "version": "6.0.16",
    "wait_command": true
  }
]
PUT https://[host]:[port]/v1/b dbs/1
200: OK
{
  ...
  "name" : "tr01",
  ...
  "slave_ha" : true,
  ...
  "uid" : 1,
  "version" : "6.0.16",
  "wait_command" : true
}

```

More info

- [Redis Enterprise Software REST API](#)
- [Redis Enterprise Software REST API requests](#)

Updated: August 17, 2023

Redis Enterprise REST API requests

A REST API request requires the following components:

- [HTTP method](#) (GET, PUT, PATCH, POST, DELETE)
- Base URL
- Endpoint

Some requests may also require:

- URL parameters
- [Query parameters](#)
- [JSON request body](#)
- [Permissions](#)

Request	Description
actions	Actions requests
b dbs	Database requests
bootstrap	Bootstrap requests
cluster	Cluster settings requests
crdb_tasks	Active-Active database task status requests
crdbs	Active-Active database requests
endpoints/stats	Endpoint statistics requests
jsonschema	API object JSON schema requests

Request	Description
ldap_mappings	LDAP mappings requests
license	License requests
logs	Cluster event logs requests
modules	Redis modules requests
nodes	Node requests
ocsp	OCSP requests
redis_acls	Redis access control list (ACL) requests
roles	Roles requests
shards/stats	Shard statistics requests
suffix	DNS suffix requests
suffixes	DNS suffixes requests
users	User requests

Updated: November 29, 2021

Actions requests

Method	Path	Description
GET	/v1/actions	Get all actions
GET	/v1/actions/{uid}	Get a single action

Get all actions

GET /v1/actions

Get the status of all actions (executing, queued, or completed) on all entities (clusters, nodes, and databases). This API tracks long-lived API requests that return either a task_id or an action_uid.

Required permissions

Permission name

view_status_of_cluster_action

Request

Example HTTP request

GET /actions

Response

Returns a JSON array of [action objects](#) and an array of [state-machine objects](#).

Regardless of an action's source, each action in the response contains the following attributes: name, action_uid, status, and progress.

Example JSON body

```
{
  "actions": [
    {
      "action_uid": "159ca2f8-7bf3-4cda-97e8-4eb560665c28",
      "name": "retry_bdb",
      "node_uid": "2",
      "progress": "100",
      "status": "completed",
      "task_id": "159ca2f8-7bf3-4cda-97e8-4eb560665c28"
    },
    {
      "action_uid": "661697c5-c747-41bd-ab81-ffc8fd13c494",
      "name": "retry_bdb",
      "node_uid": "1",
      "progress": "100",
      "status": "completed",
      "task_id": "661697c5-c747-41bd-ab81-ffc8fd13c494"
    }
  ],
  "state-machines": [
    {
      "action_uid": "a10586b1-60bc-428e-9bc6-392eb5f0d8ae",
      "heartbeat": 1650378874,
      "name": "SMCreateBDB",
      "object_name": "bdb:1",
      "progress": 100,
      "status": "completed"
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error, response provides info about an ongoing action
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run).

Get a specific action

`GET /v1/actions/{uid}`

Get the status of a currently executing, queued, or completed action.

Required permissions

Permission name

[view_status_of_cluster_action](#)

Request

Example HTTP request

`GET /actions/{uid}`

URL parameters

Field	Type	Description
-------	------	-------------

Field	Type	Description
uid	string	The action_uid to check

Response

Returns an [action object](#).

Regardless of an action's source, each action contains the following attributes: name, action_uid, status, and progress.

Example JSON body

```
{
  "action_uid": "159ca2f8-7bf3-4cda-97e8-4eb560665c28",
  "name": "retry_bdb",
  "node_uid": "2",
  "progress": "100",
  "status": "completed",
  "task_id": "159ca2f8-7bf3-4cda-97e8-4eb560665c28"
}
```

Status codes

Code	Description
200 OK	No error, response provides info about an ongoing action
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run)

Updated: April 26, 2022

Database requests

Method	Path	Description
GET	/v1/bdbs	Get all databases
GET	/v1/bdbs/{uid}	Get a single database
PUT	/v1/bdbs/{uid}	Update database configuration
PUT	/v1/bdbs/{uid}/{action}	Update database configuration and perform additional action
POST	/v1/bdbs	Create a new database
POST	/v2/bdbs	Create a new database
DELETE	/v1/bdbs/{uid}	Delete a database

Get all databases

```
GET /v1/bdbs
```

Get all databases in the cluster.

Permissions

Permission name	Roles

Permission name	Roles
view_all_b dbs_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /v1/b dbs?fields=uid,name
```

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json

Query parameters

Field	Type	Description
fields	string	Comma-separated list of field names to return (by default all fields are returned). (optional)

Response

The response body contains a JSON array with all databases, represented as [BDB objects](#).

Body

```
[
  {
    "uid": 1,
    "name": "name of database #1",
    "// additional fields..."
  },
  {
    "uid": 2,
    "name": "name of database #2",
    "// additional fields..."
  }
]
```

Status codes

Code	Description
200 OK	No error

Example requests

cURL

```
$ curl -k -X GET -u "[username]:[password]" \
-H "accept: application/json" \
https://[host][:port]/v1/b dbs?fields=uid,name
```

Python

```
import requests
import json

url = "https://[host][:port]/v1/bdbs?fields=uid,name"
auth = ("[username]", "[password]")

headers = {
    'Content-Type': 'application/json'
}

response = requests.request("GET", url, auth=auth, headers=headers)

print(response.text)
```

Get a database

```
GET /v1/bdbs/{int: uid}
```

Get a single database.

Permissions

Permission name	Roles
view_bdb_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/1
```

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database requested.

Query parameters

Field	Type	Description
fields	string	Comma-separated list of field names to return (by default all fields are returned). (optional)

Response

Returns a [BDB object](#).

Example JSON body

```
{  
  "uid": 1,  
  "name": "name of database #1",  
  "// additional fields..."  
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Database UID does not exist

Update database configuration

```
PUT /v1/b dbs/{int: uid}
```

Update the configuration of an active database.

If called with the `dry_run` URL query string, the function will validate the [BDB object](#) against the existing database, but will not invoke the state machine that will update it.

This is the basic version of the update request. See [Update database and perform action](#) to send an update request with an additional action.

To track this request's progress, poll the [/actions/<action_uid>](#) endpoint with the `action_uid` returned in the response body.

Permissions

Permission name	Roles
update_bdb	admin cluster_member db_member

Request

Example HTTP request

```
PUT /b dbs/1
```

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json
Content-type	application/json

Query parameters

Field	Type	Description
<code>dry_run</code>		Validate the new BDB object but don't apply the update.

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database for which update is requested.

Body

Include a [BDB object](#) with updated fields in the request body.

Example JSON body

```
{
  "replication": true,
  "data_persistence": "aof"
}
```

The above request attempts to modify a database configuration to enable in-memory data replication and append-only file data persistence.

Response

Returns the updated [BDB object](#).

Example JSON body

```
{
  "uid": 1,
  "replication": true,
  "data_persistence": "aof",
  "// additional fields..."
}
```

Status codes

Code	Description
200 OK	The request is accepted and is being processed. The database state will be 'active-change-pending' until the request has been fully processed.
404 Not Found	Attempting to change a nonexistent database.
406 Not Acceptable	The requested configuration is invalid.
409 Conflict	Attempting to change a database while it is busy with another configuration change. In this context, this is a temporary condition, and the request should be reattempted later.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
<code>rack_awarenessViolation</code>	<ul style="list-style-type: none"> Non rack-aware cluster. Not enough nodes in unique racks.
<code>invalid_certificate</code>	SSL client certificate is missing or malformed.
<code>certificate_expired</code>	SSL client certificate has expired.
<code>duplicated_certs</code>	An SSL client certificate appears more than once.
<code>insufficient_resources</code>	Shards count exceeds shards limit per bdb.
<code>not_supported_action_on_crdt</code>	<code>reset_admin_pass</code> action is not allowed on CRDT enabled BDB.
<code>nameViolation</code>	CRDT database name cannot be changed.
<code>bad_shards_blueprint</code>	The sharding blueprint is broken or doesn't fit the BDB.
<code>replicationViolation</code>	CRDT database must use replication.

Code	Description
eviction_policyViolation	LFU eviction policy is not supported on bdb version<4
replication_nodeViolation	Not enough nodes for replication.
replication_sizeViolation	Database limit too small for replication.
invalid_oss_cluster_configuration	BDB configuration does not meet the requirements for OSS cluster mode
missing_backup_interval	BDB backup is enabled but backup interval is missing.
crdt_shardingViolation	CRDB created without sharding cannot be changed to use sharding
invalid_proxy_policy	Invalid proxy_policy value.
invalid_bdb_tags	Tag objects with the same key parameter were passed.
unsupported_module_capabilities	Not all modules configured for the database support the capabilities needed for the database configuration.
redis_acl_unsupported	Redis ACL is not supported for this database.

Update database and perform action

```
PUT /v1/bdbs/{int: uid}/{action}
```

Update the configuration of an active database and perform an additional action.

If called with the `dry_run` URL query string, the function will validate the [BDB object](#) against the existing database, but will not invoke the state machine that will update it.

Permissions

Permission name	Roles
update_bdb_with_action	admin cluster_member db_member

Request

Example HTTP request

```
PUT /bdbs/1/reset_admin_pass
```

The above request resets the admin password after updating the database.

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json
Content-type	application/json

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database to update.
action	string	Additional action to perform. Currently supported actions are: <code>flush</code> , <code>reset_admin_pass</code> .

Query parameters

Field	Type	Description
dry_run		Validate the new BDB object but don't apply the update.

Body

Include a [BDB object](#) with updated fields in the request body.

Example JSON body

```
{
  "replication": true,
  "data_persistence": "aof"
}
```

The above request attempts to modify a database configuration to enable in-memory data replication and append-only file data persistence.

 | Note: To change the shard hashing policy, you must flush all keys from the database.

Response

If the request succeeds, the response body returns the updated [BDB object](#). If an error occurs, the response body may include an error code and message with more details.

Status codes

Code	Description
200 OK	The request is accepted and is being processed. The database state will be 'active-change-pending' until the request has been fully processed.
403 Forbidden	redislabs license expired.
404 Not Found	Attempting to change a nonexistent database.
406 Not Acceptable	The requested configuration is invalid.
409 Conflict	Attempting to change a database while it is busy with another configuration change. In this context, this is a temporary condition, and the request should be reattempted later.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
rack_awarenessViolation	<ul style="list-style-type: none"> Non rack-aware cluster. Not enough nodes in unique racks.
invalid_certificate	SSL client certificate is missing or malformed.
certificate_expired	SSL client certificate has expired.
duplicated_certs	An SSL client certificate appears more than once.
insufficient_resources	Shards count exceeds shards limit per bdb.
not_supported_action_on_crdt	<code>reset_admin_pass</code> action is not allowed on CRDT enabled BDB.
nameViolation	CRDT database name cannot be changed.
bad_shards_blueprint	The sharding blueprint is broken or doesn't fit the BDB.
replicationViolation	CRDT database must use replication.
eviction_policyViolation	LFU eviction policy is not supported on bdb version<4
replication_nodeViolation	Not enough nodes for replication.
replication_sizeViolation	Database limit too small for replication.
invalid_oss_cluster_configuration	BDB configuration does not meet the requirements for OSS cluster mode

Code	Description
missing_backup_interval	BDB backup is enabled but backup interval is missing.
crdt_shardingViolation	CRDB created without sharding cannot be changed to use sharding
invalid_proxy_policy	Invalid proxy_policy value.
invalid_bdb_tags	Tag objects with the same key parameter were passed.
unsupported_module_capabilities	Not all modules configured for the database support the capabilities needed for the database configuration.
redis_acl_unsupported	Redis ACL is not supported for this database.

Create database v1

POST /v1/bdbs

Create a new database in the cluster.

The request must contain a single JSON BDB object with the configuration parameters for the new database.

The following parameters are required to create the database:

Parameter	Type/Value	Description
name	string	Name of the new database
memory_size	integer	Size of the database, in bytes

If passed with the `dry_run` URL query string, the function will validate the [BDB object](#), but will not invoke the state machine that will create it.

To track this request's progress, poll the [/actions/<action_uid>](#) endpoint with the action_uid returned in the response body.

The cluster will use default configuration for any missing database field. The cluster creates a database UID if it is missing.

Permissions

Permission name	Roles
create_bdb	admin cluster_member db_member

Request

Example HTTP request

POST /bdbs

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json
Content-type	application/json

Query parameters

Field	Type	Description
<code>dry_run</code>		Validate the new BDB object but don't create the database.

Body

Include a [BDB object](#) in the request body.

The following parameters are required to create the database:

Paramter	Type/Value	Description
name	string	Name of the new database
memory_size	integer	Size of the database, in bytes

The uid of the database is auto-assigned by the cluster because it was not explicitly listed in this request. If you specify the database ID (id), then you must specify the database ID for every subsequent database and make sure that the database ID does not conflict with an existing database. If you do not specify the database ID, then it is automatically assigned in sequential order.

Defaults are used for all other configuration parameters.

Example JSON body

```
{  
    "name": "test-database",  
    "type": "redis",  
    "memory_size": 1073741824  
}
```

The above request is an attempt to create a Redis database with a user-specified name and a memory limit of 1GB.

Response

The response includes the newly created [BDB object](#).

Example JSON body

```
{  
    "uid": 1,  
    "name": "test-database",  
    "type": "redis",  
    "memory_size": 1073741824,  
    "// additional fields..."  
}
```

Error codes

When errors are reported, the server may return a JSON object with error_code and message field that provide additional information. The following are possible error_code values:

Code	Description
uid_exists	The specified database UID is already in use.
missing_db_name	DB name is a required property.
missing_memory_size	Memory Size is a required property.
missing_module	Modules missing from the cluster.
port_unavailable	The specified database port is reserved or already in use.
invalid_sharding	Invalid sharding configuration was specified.
bad_shards_blueprint	The sharding blueprint is broken.
not_rack_aware	Cluster is not rack-aware and a rack-aware database was requested.
invalid_version	An invalid database version was requested.
busy	The request failed because another request is being processed at the same time on the same database.
invalid_data_persistence	Invalid data persistence configuration.

Code	Description
invalid_proxy_policy	Invalid proxy_policy value.
invalid_sasl_credentials	SASL credentials are missing or invalid.
invalid_replication	Not enough nodes to perform replication.
insufficient_resources	Not enough resources in cluster to host the database.
rack_awarenessViolation	<ul style="list-style-type: none"> Rack awareness violation. Not enough nodes in unique racks.
invalid_certificate	SSL client certificate is missing or malformed.
certificate_expired	SSL client certificate has expired.
duplicated_certs	An SSL client certificate appears more than once.
replicationViolation	CRDT database must use replication.
eviction_policyViolation	LFU eviction policy is not supported on bdb version<4
invalid_oss_cluster_configuration	BDB configuration does not meet the requirements for OSS cluster mode
memcachedCannotUseModules	Cannot create a memcached database with modules.
missing_backup_interval	BDB backup is enabled but backup interval is missing.
wrong_cluster_state_id	The given CLUSTER-STATE-ID does not match the current one
invalid_bdb_tags	Tag objects with the same key parameter were passed.
unsupported_module_capabilities	Not all modules configured for the database support the capabilities needed for the database configuration.
redis_acl_unsupported	Redis ACL is not supported for this database.

Status codes

Code	Description
403 Forbidden	redislabs license expired.
409 Conflict	Database with the same UID already exists.
406 Not Acceptable	Invalid configuration parameters provided.
200 OK	Success, database is being created.

Create database v2

POST /v2/bdbs

Create a new database in the cluster. See [POST /v1/bdbs](#) for more information.

The database's configuration should be under the "bdb" field.

This endpoint allows you to specify a recovery_plan to recover a database. If you include a recovery_plan within the request body, the database will be loaded from the persistence files according to the recovery plan. The recovery plan must match the number of shards requested for the database.

The persistence files must exist in the locations specified by the recovery plan. The persistence files must belong to a database with the same shard settings as the one being created (slot range distribution and shard_key_regex); otherwise, the operation will fail or yield unpredictable results.

If you create a database with a shards_blueprint and a recovery plan, the shard placement may not fully follow the shards_blueprint.

Request

Example HTTP request

POST /v2/bdbs

Headers

Key	Value
-----	-------

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json
Content-type	application/json

Query parameters

Field	Type	Description
dry_run		Validate the new BDB object but don't create the database.

Body

Include a JSON object that contains a [BDB object](#) and an optional `recovery_plan` object in the request body.

Example JSON body

```
{
  "bdb": {
    "name": "test-database",
    "type": "redis",
    "memory_size": 1073741824,
    "shards_count": 1
  },
  "recovery_plan": {
    "data_files": [
      {
        "shard_slots": "0-16383",
        "node_uid": "1",
        "filename": "redis-4.rdb"
      }
    ]
  }
}
```

Response

The response includes the newly created [BDB object](#).

Example JSON body

```
{
  "uid": 1,
  "name": "test-database",
  "type": "redis",
  "memory_size": 1073741824,
  "shards_count": 1,
  "// additional fields..."
}
```

Delete database

```
DELETE /v1/bdbs/{int: uid}
```

Delete an active database.

Permissions

Permission name	Roles
-----------------	-------

Permission name	Roles
delete_bdb	admin cluster_member db_member

Request

Example HTTP request

```
DELETE /bdbs/1
```

Headers

Key	Value
Host	The domain name or IP of the cluster
Accept	application/json

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database to delete.

Response

Returns a status code that indicates the database deletion success or failure.

Status codes

Code	Description
200 OK	The request is accepted and is being processed. The database state will be 'delete-pending' until the request has been fully processed.
403 Forbidden	Attempting to delete an internal database.
404 Not Found	Attempting to delete a nonexistent database.
409 Conflict	Either the database is not in 'active' state and cannot be deleted, or it is busy with another configuration change. In the second case, this is a temporary condition, and the request should be re-attempted later.

Updated: August 15, 2022

Database actions requests

Backup

Method	Path	Description
PUT	/v1/bdbs/{uid}/actions/backup_reset_status	Resets database backup status

Export

Method	Path	Description
PUT	/v1/bdbs/{uid}/actions/export_reset_status	Resets database export status

Method	Path	Description
POST	/v1/b dbs/{uid}/actions/export	Initiate database export

Import

Method	Path	Description
PUT	/v1/b dbs/{uid}/actions/import_reset_status	Reset database import status
POST	/v1/b dbs/{uid}/actions/import	Initiate manual dataset import

Optimize shards placement

Method	Path	Description
GET	/v1/b dbs/{uid}/actions/optimize_shards	Get optimized shards placement for a database

Updated: November 29, 2021

Backup reset status database action requests

Method	Path	Description
PUT	/v1/b dbs/{uid}/actions/backup_reset_status	Reset database backup status

Reset database backup status

```
PUT /v1/b dbs/{int: uid}/actions/backup_reset_status
```

Resets the database's backup_status to idle if a backup is not in progress and clears the value of the backup_failure_reason field.

Permissions

Permission name	Roles
reset_bdb_current_backup_status	admin cluster_member db_member

Request

Example HTTP request

```
PUT /b dbs/1/actions/backup_reset_status
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description

Field	Type	Description
uid	integer	The unique ID of the database

Response

Returns a status code.

Status codes

Code	Description
200 OK	The request is accepted and is being processed.
404 Not Found	Attempting to perform an action on a nonexistent database.
406 Not Acceptable	Not all the modules loaded to the database support 'backup_restore' capability
409 Conflict	Database is currently busy with another action. In this context, this is a temporary condition and the request should be reattempted later.

Updated: August 3, 2022

Export database action requests

Method	Path	Description
POST	/v1/bdbs/{uid}/actions/export	Initiate database export

Initiate database export

```
POST /v1/bdbs/{int: uid}/actions/export
```

Initiate a database export.

Permissions

Permission name	Roles
start_bdb_export	admin cluster_member db_member

Request

Example HTTP request

```
POST /bdbs/1/actions/export
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database

Body

The request body should contain a JSON object with the following export parameters:

Field	Type	Description
export_location	backup_location/export_location object	Details for the export destination. Call GET /jsonschema on the bdb object and review the backup_location field to retrieve the object's structure.
email_notification	boolean	Enable/disable an email notification on export failure/ completion. (optional)

Example JSON body

```
{
  "export_location": {
    "type": "url",
    "url": "ftp://..."
  },
  "email_notification": true
}
```

The above request initiates an export operation to the specified location.

Response

Returns a status code.

Status codes

Code	Description
200 OK	The request is accepted and is being processed. In order to monitor progress, the BDB's export_status, export_progress, and export_failure_reason attributes can be consulted.
404 Not Found	Attempting to perform an action on a nonexistent database.
406 Not Acceptable	Not all the modules loaded to the database support 'backup_restore' capability
409 Conflict	Database is currently busy with another action. In this context, this is a temporary condition and the request should be reattempted later.

Updated: August 3, 2022

Export resets status database action requests

Method	Path	Description
PUT	/v1/bdbs/{uid}/actions/export_reset_status	Resets database export status

Reset database export status

```
PUT /v1/bdbs/{int: uid}/actions/export_reset_status
```

Resets the database's export_status to idle if an export is not in progress and clears the value of the export_failure_reason field.

Permissions

Permission name	Roles
reset_bdb_current_export_status	admin cluster_member db_member

Request

Example HTTP request

```
PUT /bdbss/1/actions/export_reset_status
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database

Response

Returns a status code.

Status codes

Code	Description
200 OK	The request is accepted and is being processed.
404 Not Found	Attempting to perform an action on a nonexistent database.
406 Not Acceptable	Not all the modules loaded to the database support 'backup_restore' capability
409 Conflict	Database is currently busy with another action. In this context, this is a temporary condition and the request should be reattempted later.

Updated: August 3, 2022

Import database action requests

Method	Path	Description
POST	/v1/bdbss/{uid}/actions/import	Initiate manual dataset import

Initiate manual dataset import

```
POST /v1/bdbss/{int: uid}/actions/import
```

Initiate a manual import process.

Permissions

Permission name	Roles
start_bdb_import	admin cluster_member db_member

Request

Example HTTP request

```
POST /bdb/1/actions/import
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type
Content-Length	0	Length of the request body in octets

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database

Body

The request *may* contain a subset of the [BDB JSON object](#), which includes the following import-related attributes:

Field	Type	Description
dataset_import_sources	array of dataset_import_sources objects	Details for the import sources. Call GET /jsonschema on the bdb object and review the dataset_import_sources field to retrieve the object's structure.
email_notification	boolean	Enable/disable an email notification on import failure/ completion. (optional)



Note: Other attributes are not allowed and will cause the request to fail.

Example JSON Body

```
{
  "dataset_import_sources": [
    {
      "type": "url",
      "url": "http://..."
    },
    {
      "type": "url",
      "url": "redis://..."
    }
  ],
  "email_notification": true
}
```

This request initiates an import process using dataset_import_sources values that were previously configured for the database.

Response

Returns a status code.

Status codes

Code	Description
200 OK	The request is accepted and is being processed. In order to monitor progress, the <code>import_status</code> , <code>import_progress</code> , and <code>import_failure_reason</code> attributes can be consulted.
404 Not Found	Attempting to perform an action on a nonexistent database.
406 Not Acceptable	Not all the modules loaded to the database support ‘ <code>backup_restore</code> ’ capability.
409 Conflict	Database is currently busy with another action. In this context, this is a temporary condition and the request should be reattempted later.

Updated: August 3, 2022

Import reset status database action requests

Method	Path	Description
PUT	/v1/bdbs/{uid}/actions/import_reset_status	Reset database import status

Reset database import status

PUT /v1/bdbs/{int: uid}/actions/import_reset_status

Reset the database’s `import_status` to idle if a backup is not in progress and clears the value of the `import_failure_reason` field.

Permissions

Permission name	Roles
reset_bdb_current_import_status	admin cluster_member db_member

Request

Example HTTP request

```
PUT /bdbs/1/actions/import_reset_status
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database

Response

Returns a status code.

Status codes

Code	Description
200 OK	The request is accepted and is being processed.
404 Not Found	Attempting to perform an action on a nonexistent database.
406 Not Acceptable	Not all the modules loaded to the database support 'backup_restore' capability
409 Conflict	Database is currently busy with another action. In this context, this is a temporary condition and the request should be reattempted later.

Updated: August 3, 2022

Optimize shards placement database action requests

Method	Path	Description
GET	/v1/b dbs/{uid}/actions/optimize_shards_placement	Get optimized shards placement for a database

Get optimized shards placement

```
GET /v1/b dbs/{int: uid}/actions/optimize_shards_placement
```

Get optimized shards placement for the given database.

Required permissions

Permission name	Roles
view_bdb_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /b dbs/1/actions/optimize_shards_placement
```

Query parameters

Include query parameters in a GET request to generate an optimized shard placement blueprint for a database, using settings that are different from the database's current configuration.

Field	Type	Description
avoid_nodes	list of integers	Comma-separated list of cluster node IDs to avoid when placing the database's shards and binding its endpoints (for example, avoid_nodes=1, 2)

Field	Type	Description
memory_size	integer (default: 0)	Database memory limit (0 is unlimited), expressed in bytes
shards_count	integer, (range: 1-512) (default: 1)	Number of database server-side shards
shards_placement	dense sparse	Control the density of shards dense: Shards reside on as few nodes as possible sparse: Shards reside on as many nodes as possible
bigstore_ram_size	integer (default: 0)	Memory size of bigstore RAM part, expressed in bytes

The following example request includes `shards_count` and `memory_size` as query parameters:

```
GET /bdb/1/actions/optimize_shards_placement?shards_count=10&memory_size=10000
```

Response

To rearrange the database shards, you can submit the blueprint returned in this response body as the `shards_blueprint` field in the [PUT /bdb/{uid}](#) request.

Example JSON body

```
[
  {
    "nodes": [
      {
        "node_uid": "3",
        "role": "master"
      },
      {
        "node_uid": "1",
        "role": "slave"
      }
    ],
    "slot_range": "5461-10922"
  },
  {
    "nodes": [
      {
        "node_uid": "3",
        "role": "master"
      },
      {
        "node_uid": "1",
        "role": "slave"
      }
    ],
    "slot_range": "10923-16383"
  },
  {
    "nodes": [
      {
        "node_uid": "3",
        "role": "master"
      },
      {
        "node_uid": "1",
        "role": "slave"
      }
    ],
    "slot_range": "0-5460"
  }
]
```

Headers

Key	Value	Description
Content-Length	352	Length of the request body in octets
cluster-state-id	30	Cluster state ID

Status codes

Code	Description
200 OK	No error
404 Not Found	Database UID does not exist
406 Not Acceptable	Not enough resources in the cluster to host the database

Rearrange database shards

Use the blueprint returned by the `GET /bdbss/{uid}/actions/optimize_shards_placement` request as the value of the `shards_blueprint` field to rearrange the database shards.

To ensure that the optimized shard placement is relevant for the current cluster state, pass the `cluster-state-id`, taken from the response header of the GET request, in the `PUT /bdbss/{uid}` request headers.

The cluster will reject the update if its state was changed since the optimal shards placement was obtained.

Request

Example HTTP request

```
PUT /bdbss/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type
cluster-state-id	30	Cluster state ID

Example JSON body

```
{
  "shards_blueprint": [
    {
      "nodes": [
        {
          "node_uid": "2",
          "role": "master"
        }
      ],
      "slot_range": "0-8191"
    },
    ...
  ]
}
```



Warning - If you submit such an optimized blueprint, it may cause strain on the cluster and its resources. Use with caution.

Database alerts requests

Method	Path	Description
GET	/v1/bdbs/alerts	Get all alert states for all databases
GET	/v1/bdbs/alerts/{uid}	Get all alert states for a specific database
GET	/v1/bdbs/alerts/{uid}/{alert}	Get a specific database alert state
POST	/v1/bdbs/alerts/{uid}	Update a database's alerts configuration

Get all database alerts

```
GET /bdbs/alerts
```

Get all alert states for all databases.

Required permissions

Permission name

view_all_bdbs_alerts

Request

Example HTTP request

```
GET /bdbs/alerts
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a hash of alert UIDs and the `alerts` states for each database.

Example JSON body

```
{
  "1": {
    "bdb_size": {
      "enabled": true,
      "state": true,
      "threshold": "80",
      "change_time": "2014-08-29T11:19:49Z",
      "severity": "WARNING",
      "change_value": {
        "state": true,
        "threshold": "80",
        "memory_util": 81.2
      }
    },
    "..."
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error

Get database alerts

GET /v1/bdbs/alerts/{int: uid}

Get all alert states for a database.

Required permissions

Permission name

[view_bdb_alerts](#)

Request

Example HTTP request

GET /bdbs/alerts/1

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{
  "bdb_size": {
    "enabled": true,
    "state": true,
    "threshold": "80",
    "severity": "WARNING",
    "change_time": "2014-08-29T11:19:49Z",
    "change_value": {
      "state": true,
      "threshold": "80",
      "memory_util": 81.2
    }
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified bdb does not exist

Get database alert

GET /v1/bdbs/alerts/{int: uid}/{alert}

Get a database alert state.

Required permissions

Permission name

view_bdb_alerts

Request

Example HTTP request

GET /bdbs/alerts/1/bdb_size

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database
alert	string	The alert name

Response

Returns an [alert object](#).

Example JSON body

```
{  
    "enabled": true,  
    "state": true,  
    "threshold": "80",  
    "severity": "WARNING",  
    "change_time": "2014-08-29T11:19:49Z",  
    "change_value": {  
        "state": true,  
        "threshold": "80",  
        "memory_util": 81.2  
    }  
}
```

Status codes

Code	Description
200 OK	No error
400 Bad Request	Bad request
404 Not Found	Specified alert or bdb does not exist

Update database alert

POST /v1/b dbs/alerts/{int: uid}

Updates a database's alerts configuration.

Required permissions

Permission name

[update_bdb_alerts](#)

Request

If passed with the dry_run URL query string, the function will validate the alert thresholds, but not commit them.

Example HTTP request

POST /b dbs/alerts/1

Example JSON body

```
{  
    "bdb_size": {  
        "threshold": "80",  
        "enabled": true  
    },  
    "bdb_high_syncer_lag": {  
        "threshold": "",  
        "enabled": false  
    },  
    "bdb_low_throughput": {  
        "threshold": "1",  
        "enabled": true  
    },  
    "bdb_high_latency": {  
        "threshold": "3000",  
        "enabled": true  
    },  
    "bdb_high_throughput": {  
        "threshold": "1",  
        "enabled": true  
    },  
    "bdb_backup_delayed": {  
        "threshold": "1800",  
        "enabled": true  
    }  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	Database ID

Field	Type	Description
dry_run	string	Validate the alert thresholds but do not apply them

Request body

The request must contain a single JSON object with one or many database [alert objects](#).

Response

The response includes the updated database [alerts](#).

Status codes

Code	Description
404 Not Found	Specified database was not found.
406 Not Acceptable	Invalid configuration parameters provided.
200 OK	Success, database alerts updated.

Updated: November 29, 2021

Database CRDT sources alerts requests

Method	Path	Description
GET	/v1/bdbs/crdt_sources/alerts	Get all CRDT sources alert states for all CRDB databases
GET	/v1/bdbs/crdt_sources/alerts/{uid}	Get all CRDT sources alert states for a database
GET	/v1/bdbs/crdt_sources/alerts/{uid}/{crdt_source_id}	Get alert states for a CRDT source
GET	/v1/bdbs/crdt_sources/alerts/{uid}/{crdt_source_id}/{database}/[alertstate]	Get database alert state

Get all CRDB CRDT source alert states

```
GET /v1/bdbs/crdt_sources/alerts
```

Get all alert states for all CRDT sources of all CRDBs.

Required permissions

Permission name

[view_all_bdbs_alerts](#)

Request

Example HTTP request

```
GET /bdbs/crdt_sources/alerts
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a hash of alert UIDs and the [alerts states](#) for each local BDB of CRDB.

Example JSON body

```
{  
    "1": {  
        "crdt_src_syncer_connection_error": {  
            "enabled": true,  
            "state": true,  
            "threshold": "80",  
            "change_time": "2014-08-29T11:19:49Z",  
            "severity": "WARNING",  
            "change_value": {  
                "state": true,  
                "threshold": "80",  
                "memory_util": 81.2  
            }  
        },  
        "..."  
    },  
    "..."  
}
```

Status codes

Code	Description
200 OK	No error

Get all BDB CRDT sources alert states

```
GET /v1/bdbs/crdt_sources/alerts/{int: uid}
```

Get all alert states for all crdt sources for a specific local bdb of a CRDB.

Required permissions

Permission name
view_bdb_alerts

Request

Example HTTP request

```
GET /bdbs/crdt_sources/alerts/1
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description

Field	Type	Description
uid	integer	The unique ID of the database

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{
  "crdt_src_syncer_connection_error": {
    "enabled": true,
    "state": true,
    "threshold": "80",
    "severity": "WARNING",
    "change_time": "2014-08-29T11:19:49Z",
    "change_value": {
      "state": true,
      "threshold": "80",
      "memory_util": 81.2
    }
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified bdb does not exist

Get all CRDT source alert states

```
GET /v1/bdbs/crdt_sources/alerts/{int: uid}/{int: crdt_src_id}
```

Get all alert states for specific crdt source for a specific local BDB of a CRDB.

Required permissions

Permission name

[view_bdb_alerts](#)

Request

Example HTTP request

```
GET /bdbs/crdt_sources/alerts/1/2
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database
crdt_src_id	integer	The ID of the crdt source in this BDB

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{
  "crdt_src_syncer_connection_error": {
    "enabled": true,
    "state": true,
    "threshold": "80",
    "severity": "WARNING",
    "change_time": "2014-08-29T11:19:49Z",
    "change_value": {
      "state": true,
      "threshold": "80",
      "memory_util": 81.2
    }
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified bdb does not exist

Get database alert state

```
GET /v1/bdbs/crdt_sources/alerts/{int: uid}/{int: crdt_src_id}/{alert}
```

Get a BDB alert state.

Required permissions

Permission name

[view_bdb_alerts](#)

Request

Example HTTP request

```
GET /bdbs/crdt_sources/alerts/1/2/crdt_src_syncer_connection_error
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database
crdt_src_id	integer	The ID of the crdt source in this BDB
alert	string	The alert name

Response

Returns an [alert object](#).

Example JSON body

```
{
    "enabled": true,
    "state": true,
    "threshold": "80",
    "severity": "WARNING",
    "change_time": "2014-08-29T11:19:49Z",
    "change_value": {
        "state": true,
        "threshold": "80",
        "memory_util": 81.2
    }
}
```

Status codes

Code	Description
200 OK	No error
400 Bad Request	Bad request
404 Not Found	Specified alert or bdb does not exist

Updated: November 29, 2021

Database modules requests

Configure module

Method	Path	Description
POST	/v1/bdbs/{uid}/modules/config	Configure module

Upgrade module

Method	Path	Description
POST	/v1/bdbs/{uid}/modules/upgrade	Upgrade module

Updated: November 29, 2021

Database modules config requests

Method	Path	Description
--------	------	-------------

Method	Path	Description
POST	/v1/bdbs/{uid}/modules/config	Configure module

Configure module

POST /v1/bdbs/{string: uid}/modules/config

Use the module runtime configuration command (if defined) to configure new arguments for the module.

Required permissions

Permission name

[edit_bdb_module](#)

Request

Example HTTP request

POST /bdbs/1/modules/config

Example JSON body

```
{
  "modules": [
    {
      "module_name": "search",
      "module_args": "MINPREFIX 3 MAXEXPANSIONS 1000"
    }
  ]
}
```

Request headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Field	Type	Description
modules	list of JSON objects	List of modules (module_name) and their new configuration settings (module_args)
module_name	search ReJSON graph timeseries bf	Module's name
module_args	string	Module command line arguments (pattern does not allow special characters &,<,>,")

Response

Returns a status code. If an error occurs, the response body may include an error code and message with more details.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
db_not_exist	Database with given UID doesn't exist in cluster
missing_field	"module_name" or "module_args" are not defined in request
invalid_schema	JSON object received is not a dict object
param_error	"module_args" parameter was not parsed properly
module_not_exist	Module with given "module_name" does not exist for the database

Status codes

Code	Description
200 OK	Success, module updated on bdb.
404 Not Found	bdb not found.
400 Bad Request	Bad or missing configuration parameters.
406 Not Acceptable	Module does not support runtime configuration of arguments.

Updated: February 23, 2023

Database upgrade modules requests

Method	Path	Description
POST	/v1/bdbs/{uid}/modules/upgrade	Upgrade module

Upgrade module

POST /v1/bdbs/{string: uid}/modules/upgrade

Upgrades module version on a specific BDB.

Required permissions

Permission name

[edit_bdb_module](#)

Request

Example HTTP request

POST /bdbs/1/modules/upgrade

Example JSON body

```
{
  "modules": [
    {"module_name": "ReJson",
     "current_semantic_version": "2.2.1",
     "new_module": "aa3648d79bd4082d414587c42ea0b234"}
  ],
  // Optional fields to fine-tune restart and failover behavior:
  "preserve_roles": true,
  "may_discard_data": false
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Field	Type	Description
modules	list	<p>List of dicts representing the modules that will be upgraded. Each dict must include:</p> <ul style="list-style-type: none"> • current_module: uid of a module to upgrade • new_module: UID of the module we want to upgrade to • new_module_args: args list for the new module
preserve_roles	boolean	Preserve shards' master/replica roles (optional)
may_discard_data	boolean	Discard data in a non-replicated non-persistent bdb (optional)

Response

Returns the upgraded [module object](#).

Example JSON body

```
{
  "uid": 1,
  "name": "name of database #1",
  "module_id": "aa3648d79bd4082d414587c42ea0b234",
  "module_name": "ReJson",
  "semantic_version": "2.2.2"
  // additional fields...
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
missing_module	Module is not present in cluster.
module_downgrade_unsupported	Module downgrade is not allowed.
redis_incompatible_version	Module min_redis_version is bigger than the current Redis version.

Code	Description
redis_pack_incompatible_version	Module min_redis_pack_version is bigger than the current Redis Enterprise version.
unsupported_module_capabilities	New version of module does support all the capabilities needed for the database configuration

Status codes

Code	Description
200 OK	Success, module updated on bdb.
404 Not Found	bdb or node not found.
400 Bad Request	Bad or missing configuration parameters.
406 Not Acceptable	The requested configuration is invalid.

Updated: November 29, 2021

Database passwords requests

Method	Path	Description
PUT	/v1/bdbs/{uid}/passwords	Update database password
POST	/v1/bdbs/{uid}/passwords	Add database password
DELETE	/v1/bdbs/{uid}/passwords	Delete database password

Update database password

PUT /v1/bdbs/{int: uid}/passwords

Set a single password for the bdb's default user (i.e., for AUTH <password> authentications).

Required permissions

Permission name

update_bdb

Request

Example HTTP request

PUT /bdbs/1/passwords

Example JSON body

```
{
  "password": "new password"
}
```

The above request resets the password of the bdb to 'new password'.

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database to update the password.

Request body

Field	Type	Description
password	string	New password

Response

Returns a status code that indicates password update success or failure.

Status codes

Code	Description
200 OK	The password was changed.
404 Not Found	A nonexistent database.
406 Not Acceptable	Invalid configuration parameters provided.

Add database password

POST /v1/bdbs/{int: uid}/passwords

Add a password to the bdb's default user (i.e., for AUTH <password> authentications).

Required permissions

Permission name

update_bdb

Request

Example HTTP request

POST /bdbs/1/passwords

Example JSON body

```
{
  "password": "password_to_add"
}
```

The above request adds a password to the bdb.

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database to add password.

Request body

Field	Type	Description
password	string	Password to add

Response

Returns a status code that indicates password creation success or failure.

Status codes

Code	Description
200 OK	The password was added.
404 Not Found	A nonexistent database.
406 Not Acceptable	Invalid configuration parameters provided.

Delete database password

```
DELETE /v1/bdbs/{int: uid}/passwords
```

Delete a password from the bdb's default user (i.e., for AUTH <password> authentications).

Required permissions

Permission name

update_bdb

Request

Example HTTP request

```
DELETE /bdbs/1/passwords
```

Example JSON body

```
{
  "password": "password to delete"
}
```

The above request deletes a password from the bdb.

Request headers

Key	Value	Description
-----	-------	-------------

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database to delete password.

Request body

Field	Type	Description
password	string	Password to delete

Response

Returns a status code that indicates password deletion success or failure.

Status codes

Code	Description
200 OK	The password was deleted.
404 Not Found	A nonexistent database.
406 Not Acceptable	Invalid configuration parameters provided.

Updated: November 29, 2021

CRDB peer stats requests

Method	Path	Description
GET	/v1/b dbs/{bdb_uid}/peer_stats	Get stats for all CRDB peer instances
GET	/v1/b dbs/{bdb_uid}/peer_stats/{uid}	Get stats for a specific CRDB peer instance

Get all CRDB peer stats

```
GET /v1/b dbs/{bdb_uid}/peer_stats
```

Get statistics for all peer instances of a local CRDB instance.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbss/1/peer_stats?interval=5min
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
bdb_uid	integer	The unique ID of the local CRDB instance.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for all CRDB peer instances.

Example JSON body

```
{
  "peer_stats": [
    {
      "intervals": [
        {
          "egress_bytes": 0.0,
          "egress_bytes_decompressed": 0.0,
          "etime": "2017-10-22T19:30:00Z",
          "ingress_bytes": 18528,
          "ingress_bytes_decompressed": 185992,
          "interval": "5min",
          "local_ingress_lag_time": 0.244,
          "pending_local_writes_max": 0.0,
          "pending_local_writes_min": 0.0,
          "stime": "2017-10-22T19:25:00Z"
        },
        {
          "egress_bytes": 0.0,
          "egress_bytes_decompressed": 0.0,
          "etime": "2017-10-22T19:35:00Z",
          "ingress_bytes": 18,
          "ingress_bytes_decompressed": 192,
          "interval": "5min",
          "local_ingress_lag_time": 0.0,
          "pending_local_writes_max": 0.0,
          "pending_local_writes_min": 0.0,
          "stime": "2017-10-22T19:30:00Z"
        }
      ],
      "uid": "3"
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Database does not exist.
406 Not Acceptable	Database is not a CRDB.

Get CRDB peer stats

```
GET /v1/bdbs/{bdb_uid}/peer_stats/{int: uid}
```

Get statistics for a specific CRDB peer instance.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/1/peer_stats/3?interval=5min
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
bdb_uid	integer	The unique ID of the local CRDB instance.
uid	integer	The peer instance uid, as specified in the CRDB instance list.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for a specific CRDB peer instance.

Example JSON body

```
{
  "intervals": [
    {
      "egress_bytes": 0.0,
      "egress_bytes_decompressed": 0.0,
      "etime": "2017-10-22T19:30:00Z",
      "ingress_bytes": 18528,
      "ingress_bytes_decompressed": 185992,
      "interval": "5min",
      "local_ingress_lag_time": 0.244,
      "pending_local_writes_max": 0.0,
      "pending_local_writes_min": 0.0,
      "stime": "2017-10-22T19:25:00Z"
    },
    {
      "egress_bytes": 0.0,
      "egress_bytes_decompressed": 0.0,
      "etime": "2017-10-22T19:35:00Z",
      "ingress_bytes": 18,
      "ingress_bytes_decompressed": 192,
      "interval": "5min",
      "local_ingress_lag_time": 0.0,
      "pending_local_writes_max": 0.0,
      "pending_local_writes_min": 0.0,
      "stime": "2017-10-22T19:30:00Z"
    }
  ],
  "uid": "3"
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Database or peer does not exist.
406 Not Acceptable	Database is not a CRDB.

Updated: August 3, 2022

Database replica sources alerts requests

Method	Path	Description
GET	/v1/bdbs/replica_sources/alerts	Get all replica sources alert states for all BDBs
GET	/v1/bdbs/replica_sources/alerts/{uid}	Get all replica sources alert states for a BDB
GET	/v1/bdbs/replica_sources/alerts/{uid} {Get alert states for a replica source}	
GET	/v1/bdbs/replica_sources/alerts/{uid} {Get replica source alert}	

Get all DBs replica sources alert states

GET /v1/bdbs/replica_sources/alerts

Get all alert states for all replica sources of all BDBs.

Required permissions

Permission name

view_all_bdbs_alerts

Request

Example HTTP request

GET /bdbs/replica_sources/alerts

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a hash of alert UIDs and the alerts states for each BDB.

See [REST API alerts overview] for a description of the alert state object.

Example JSON body

```
{
    "1": {
        "replica_src_syncer_connection_error": {
            "enabled": true,
            "state": true,
            "threshold": "80",
            "change_time": "2014-08-29T11:19:49Z",
            "severity": "WARNING",
            "change_value": {
                "state": true,
                "threshold": "80",
                "memory_util": 81.2
            }
        },
        "..."
    },
    ...
}
```

Status codes

Code	Description
200 OK	No error

Get DB replica source alert states

GET /v1/bdbs/replica_sources/alerts/{int: uid}

Get all alert states for all replica sources of a specific bdb.

Required permissions

Permission name

view_bdb_alerts

Request

Example HTTP request

GET /bdbs/replica_sources/alerts/1

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{  
    "replica_src_syncer_connection_error": {  
        "enabled": true,  
        "state": true,  
        "threshold": "80",  
        "severity": "WARNING",  
        "change_time": "2014-08-29T11:19:49Z",  
        "change_value": {  
            "state": true,  
            "threshold": "80",  
            "memory_util": 81.2  
        }  
    },  
    "..."  
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified bdb does not exist

Get replica source alert states

GET /v1/bdbs/replica_sources/alerts/{int: uid}/{int: replica_src_id}

Get all alert states for a specific replica source of a bdb.

Required permissions

Permission name
view_bdb_alerts

Request

Example HTTP request

GET /bdbs/replica_sources/alerts/1/2

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database
replica_src_id	integer	The ID of the replica source in this BDB

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{  
    "replica_src_syncer_connection_error": {  
        "enabled": true,  
        "state": true,  
        "threshold": "80",  
        "severity": "WARNING",  
        "change_time": "2014-08-29T11:19:49Z",  
        "change_value": {  
            "state": true,  
            "threshold": "80",  
            "memory_util": 81.2  
        }  
    },  
    "..."  
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified bdb does not exist

Get replica source alert state

```
GET /v1/bdbs/replica_sources/alerts/{int: uid}/{int: replica_src_id}/{alert}
```

Get a replica source alert state of a specific bdb.

Required permissions

Permission name
view_bdb_alerts

Request

Example HTTP request

```
GET /bdbs/replica_sources/alerts/1/2/replica_src_syncer_connection_error
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the database
replica_src_id	integer	The ID of the replica source in this BDB
alert	string	The alert name

Response

Returns an alert state object.

Example JSON body

```
{  
    "enabled": true,  
    "state": true,  
    "threshold": "80",  
    "severity": "WARNING",  
    "change_time": "2014-08-29T11:19:49Z",  
    "change_value": {  
        "state": true,  
        "threshold": "80",  
        "memory_util": 81.2  
    }  
}
```

Status codes

Code	Description
200 OK	No error
400 Bad Request	Bad request
404 Not Found	Specified alert or bdb does not exist

Updated: November 29, 2021

Database stats requests

Method	Path	Description
GET	/v1/bdbs/stats	Get stats for all databases
GET	/v1/bdbs/stats/{uid}	Get stats for a specific database

Get all database stats

```
GET /v1/bdbs/stats
```

Get statistics for all databases.

Permissions

Permission name	Roles
view_all_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/stats?interval=1hour&stime=2014-08-28T10:00:00Z
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for all databases.

Example JSON body

```
[
  {
    "uid": "1",
    "intervals": [
      {
        "interval": "1hour",
        "stime": "2015-05-27T12:00:00Z",
        "etime": "2015-05-28T12:59:59Z",
        "avg_latency": 0.0,
        "conns": 0.0,
        "egress_bytes": 0.0,
        "etime": "2015-05-28T00:00:00Z",
        "evicted_objects": 0.0,
        "expired_objects": 0.0,
        "ingress_bytes": 0.0,
        "instantaneous_ops_per_sec": 0.00011973180076628352,
        "last_req_time": "1970-01-01T00:00:00Z",
        "last_res_time": "1970-01-01T00:00:00Z",
        "used_memory": 5656299.362068966,
        "mem_size_lua": 35840.0,
        "monitor_sessions_count": 0.0,
        "no_of_keys": 0.0,
        "other_req": 0.0,
        "other_res": 0.0,
        "read_hits": 0.0,
        "read_misses": 0.0,
        "read_req": 0.0,
        "read_res": 0.0,
        "total_connections_received": 0.0,
        "total_req": 0.0,
        "total_res": 0.0,
        "write_hits": 0.0,
        "write_misses": 0.0,
        "write_req": 0.0,
        "write_res": 0.0
      },
      {
        "interval": "1hour",
        "interval": "1hour",
        "stime": "2015-05-27T13:00:00Z",
        "etime": "2015-05-28T13:59:59Z",
        "avg_latency": 599.08,
        "// additional fields..."
      }
    ]
  },
  {
    "uid": "2",
    "intervals": [
      {
        "interval": "1hour",
        "stime": "2015-05-27T12:00:00Z",
        "etime": "2015-05-28T12:59:59Z",
        "avg_latency": 0.0,
        "// additional fields..."
      },
      {
        "interval": "1hour",
        "stime": "2015-05-27T13:00:00Z",
        "etime": "2015-05-28T13:59:59Z",
        "// additional fields..."
      }
    ]
  }
]
```

Status codes

Code	Description
200 OK	No error
404 Not Found	No bdb exist

Example requests

cURL

```
$ curl -k -u "[username]:[password]" -X GET  
https://[host][:port]/v1/bdbs/stats?interval=1hour
```

Python

```
import requests  
  
url = "https://[host][:port]/v1/bdbs/stats?interval=1hour"  
auth = ("[username]", "[password]")  
  
response = requests.request("GET", url, auth=auth)  
  
print(response.text)
```

Get database stats

```
GET /v1/bdbs/stats/{int: uid}
```

Get statistics for a specific database.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/stats/1?interval=1hour&stime=2014-08-28T10:00:00Z
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the BDB requested.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for a specific database.

Example JSON body

```
{
  "uid": "1",
  "intervals": [
    {
      "interval": "1hour",
      "stime": "2015-05-27T12:00:00Z",
      "etime": "2015-05-28T12:59:59Z",
      "avg_latency": 0.0,
      "conns": 0.0,
      "egress_bytes": 0.0,
      "evicted_objects": 0.0,
      "pubsub_channels": 0,
      "pubsub_patterns": 0,
      "expired_objects": 0.0,
      "ingress_bytes": 0.0,
      "instantaneous_ops_per_sec": 0.00011973180076628352,
      "last_req_time": "1970-01-01T00:00:00Z",
      "last_res_time": "1970-01-01T00:00:00Z",
      "used_memory": 5656299.362068966,
      "mem_size_lua": 35840.0,
      "monitor_sessions_count": 0.0,
      "no_of_keys": 0.0,
      "other_req": 0.0,
      "other_res": 0.0,
      "read_hits": 0.0,
      "read_misses": 0.0,
      "read_req": 0.0,
      "read_res": 0.0,
      "total_connections_received": 0.0,
      "total_req": 0.0,
      "total_res": 0.0,
      "write_hits": 0.0,
      "write_misses": 0.0,
      "write_req": 0.0,
      "write_res": 0.0
    },
    {
      "interval": "1hour",
      "stime": "2015-05-27T13:00:00Z",
      "etime": "2015-05-28T13:59:59Z",
      "// additional fields..."
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	bdb does not exist
406 Not Acceptable	bdb isn't currently active
503 Service Unavailable	bdb is in recovery state

Updated: August 3, 2022

Latest database stats requests

Method	Path	Description
GET	/v1/bdbs/stats/last	Get most recent stats for all databases
GET	/v1/bdbs/{uid}	Get most recent stats for a specific database

Get latest stats for all databases

```
GET /v1/bdbs/stats/last
```

Get the most recent statistics for all databases.

Required permissions

Permission name	Roles
view_all_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

- Without metrics filter (returns all metrics by default)

```
GET /bdbs/stats/last
```

- With metrics filter

```
GET /bdbs/stats/last?metrics=no_of_keys,used_memory
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
-------	------	-------------

Field	Type	Description
metrics	string	Comma-separated list of metric names for which we want statistics (default is all). (optional)

Response

Returns [statistics](#) for all databases.

Example JSON body

- Without metrics filter (returns all metrics by default)

```
{
  "1": {
    "stime": "2015-05-28T08:06:37Z",
    "etime": "2015-05-28T08:06:44Z",
    "conns": 0.0,
    "egress_bytes": 0.0,
    "etime": "2015-05-28T08:06:44Z",
    "evicted_objects": 0.0,
    "expired_objects": 0.0,
    "ingress_bytes": 0.0,
    "instantaneous_ops_per_sec": 0.0,
    "last_req_time": "1970-01-01T00:00:00Z",
    "last_res_time": "1970-01-01T00:00:00Z",
    "used_memory": 5651336.0,
    "mem_size_lua": 35840.0,
    "monitor_sessions_count": 0.0,
    "no_of_keys": 0.0,
    "other_req": 0.0,
    "other_res": 0.0,
    "read_hits": 0.0,
    "read_misses": 0.0,
    "read_req": 0.0,
    "read_res": 0.0,
    "total_connections_received": 0.0,
    "total_req": 0.0,
    "total_res": 0.0,
    "write_hits": 0.0,
    "write_misses": 0.0,
    "write_req": 0.0,
    "write_res": 0.0
  },
  "2": {
    "stime": "2015-05-28T08:06:37Z",
    "etime": "2015-05-28T08:06:44Z",
    "// additional fields..."
  },
  "// Additional BDBs..."
}
```

- With metrics filter

```

{
  "1": {
    "etime": "2015-05-28T08:06:44Z",
    "used_memory": 5651576.0,
    "no_of_keys": 0.0,
    "stime": "2015-05-28T08:06:37Z"
  },
  "2": {
    "etime": "2015-05-28T08:06:44ZZ",
    "used_memory": 5651440.0,
    "no_of_keys": 0.0,
    "stime": "2015-05-28T08:06:37Z"
  },
  "// Additional BDBs.."
}

```

Status codes

Code	Description
200 OK	No error
404 Not Found	No bdb exist

Get latest database stats

```
GET /v1/bdbs/stats/last/{int: uid}
```

Get the most recent statistics for a specific database.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/stats/last/1?metrics=no_of_keys,used_memory
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the requested BDB.

Query parameters

Field	Type	Description
metrics	string	Comma-separated list of metric names for which we want statistics (default is all). (optional)

Response

Returns the most recent [statistics](#) for a specific database.

Example JSON body

```
{
  "1": {
    "etime": "2015-06-23T12:05:08Z",
    "used_memory": 5651576.0,
    "no_of_keys": 0.0,
    "stime": "2015-06-23T12:05:03Z"
  }
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	bdb does not exist
406 Not Acceptable	bdb isn't currently active
503 Service Unavailable	bdb is in recovery state

Updated: August 3, 2022

Database syncer source stats requests

Method	Path	Description
GET	/v1/bdbs/{bdb_uid}/sync_source_stats	Get stats for all syncer sources
GET	/v1/bdbs/{bdb_uid}/sync_source_stats/{ GetId }	Get stats for a specific syncer instance

Get all syncer source stats

```
GET /v1/bdbs/{bdb_uid}/sync_source_stats
```

Get stats for all syncer sources of a local database.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbss/1/sync_source_stats?interval=5min
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
bdb_uid	integer	The unique ID of the local database.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	Optional end time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for all syncer sources.

Example JSON body

```
{
  "sync_source_stats": [
    {
      "intervals": [
        {
          "etime": "2017-10-22T19:30:00Z",
          "ingress_bytes": 18528,
          "ingress_bytes_decompressed": 185992,
          "interval": "5min",
          "local_ingress_lag_time": 0.244,
          "stime": "2017-10-22T19:25:00Z"
        },
        {
          "etime": "2017-10-22T19:35:00Z",
          "ingress_bytes": 18,
          "ingress_bytes_decompressed": 192,
          "interval": "5min",
          "local_ingress_lag_time": 0.0,
          "stime": "2017-10-22T19:30:00Z"
        }
      ],
      "uid": "1"
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Database does not exist.

Get syncer instance stats

```
GET /v1/bdbs/{bdb_uid}/sync_source_stats/{int: uid}
```

Get stats for a specific syncer (Replica Of) instance.

Permissions

Permission name	Roles
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /bdbs/1/sync_source_stats/1?interval=5min
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
bdb_uid	integer	The unique ID of the local database.
uid	integer	The sync_source uid.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Optional start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	Optional end time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for a specific syncer instance.

Example JSON body

```
{
  "intervals": [
    {
      "etime": "2017-10-22T19:30:00Z",
      "ingress_bytes": 18528,
      "ingress_bytes_decompressed": 185992,
      "interval": "5min",
      "local_ingress_lag_time": 0.244,
      "stime": "2017-10-22T19:25:00Z"
    },
    {
      "etime": "2017-10-22T19:35:00Z",
      "ingress_bytes": 18,
      "ingress_bytes_decompressed": 192,
      "interval": "5min",
      "local_ingress_lag_time": 0.0,
      "stime": "2017-10-22T19:30:00Z"
    }
  ],
  "uid": "1"
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Database or sync_source do not exist.

Updated: August 3, 2022

Database upgrade requests

Method	Path	Description
POST	/v1/bdbs/{uid}/upgrade	Upgrade database

Upgrade database

POST /v1/bdbs/{int: uid}/upgrade

Upgrade a database.

Required permissions

Permission name
update_bdb_with_action

Request

Example HTTP request

POST /bdbs/1/upgrade

Example JSON body

```
{
  "swap_roles": true,
  "may_discard_data": false
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Field	Type	Description
force_restart	boolean	Restart shards even if no version change (default: false)
keep_redis_version	boolean	Keep current Redis version (default: false)
keep_crdt_protocol_version	boolean	Keep current crdt protocol version (default: false)
may_discard_data	boolean	Discard data in a non-replicated, non-persistent bdb (default: false)
force_discard	boolean	Discard data even if the bdb is replicated and/or persistent (default: false)
preserve_roles	boolean	Preserve shards' master/replica roles (requires an extra failover) (default: false)
parallel_shards_upgrade	integer	Max number of shards to upgrade in parallel (default: all)
modules	list of modules	<p>List of dicts representing the modules that will be upgraded.</p> <p>Each dict includes:</p> <ul style="list-style-type: none"> • current_module: uid of a module to upgrade • new_module: uid of the module we want to upgrade to • new_module_args: args list for the new module (no defaults for the three module-related parameters).

Response

Returns the upgraded [BDB object](#).

Example JSON body

```
{
  "uid": 1,
  "replication": true,
  "data_persistence": "aof",
  "// additional fields..."
}
```

Status codes

Code	Description
200 OK	Success, bdb upgrade initiated (<code>action_uid</code> can be used to track progress)

Code	Description
400 Bad Request	Malformed or bad command
404 Not Found	bdb not found
406 Not Acceptable	New module version capabilities don't comply with the database configuration
500 Internal Server Error	Internal error

Updated: November 29, 2021

Bootstrap requests

Method	Path	Description
GET	/v1/bootstrap	Get the local node's bootstrap status
POST	/v1/bootstrap/{action}	Initiate bootstrapping

Get bootstrap status

```
GET /v1/bootstrap
```

Get the local node's bootstrap status.

This request is accepted as soon the cluster software is installed and before the node is part of an active cluster.

Once the node is part of an active cluster, authentication is required.

Request

Example HTTP request

```
GET /bootstrap
```

Headers

Key	Value	Description
Accept	application/json	Accepted media type

Response

The JSON response object contains a `bootstrap_status` object and a `local_node_info` object.

The `bootstrap_status` object contains the following information:

Field	Description
	Current bootstrap state. <code>idle</code> : No bootstrapping started. <code>initiated</code> : Bootstrap request received. <code>creating_cluster</code> : In the process of creating a new cluster. <code>joining_cluster</code> : In the process of joining an existing cluster. <code>error</code> : The last bootstrap action failed. <code>completed</code> : The last bootstrap action completed successfully.
state	

Field	Description
start_time	Bootstrap process start time
end_time	Bootstrap process end time
error_code	If state is error, this error code describes the type of error encountered.
error_details	An error-specific object that may contain additional information about the error. A common field in use is message which provides a more verbose error message.

The local_node_info object is a subset of a [node object](#) that provides information about the node configuration.

Example JSON body

```
{
  "bootstrap_status": {
    "start_time": "2014-08-29T11:19:49Z",
    "end_time": "2014-08-29T11:19:49Z",
    "state": "completed"
  },
  "local_node_info": {
    "uid": 3,
    "software_version": "0.90.0-1",
    "cores": 2,
    "ephemeral_storage_path": "/var/opt/redislabs/tmp",
    "ephemeral_storage_size": 1018889.8304,
    "os_version": "Ubuntu 14.04 LTS",
    "persistent_storage_path": "/var/opt/redislabs/persist/redis",
    "persistent_storage_size": 1018889.8304,
    "total_memory": 24137,
    "uptime": 50278,
    "available_addrs": [
      {
        "address": "172.16.50.122",
        "format": "ipv4",
        "if_name": "eth0",
        "private": true
      },
      {
        "address": "10.0.3.1",
        "format": "ipv4",
        "if_name": "lxcbr0",
        "private": true
      },
      {
        "address": "172.17.0.1",
        "format": "ipv4",
        "if_name": "docker0",
        "private": true
      },
      {
        "address": "2001:db8:0:f101::1",
        "format": "ipv6",
        "if_name": "eth0",
        "private": false
      }
    ]
  }
}
```

Error codes

Code	Description
config_error	An error related to the bootstrap configuration provided (e.g. bad JSON).
connect_error	Failed to connect to cluster (e.g. FQDN DNS could not resolve, no/wrong node IP provided, etc.).

Code	Description
access_denied	Invalid credentials supplied.
invalid_license	The license string provided is invalid. Additional info can be fetched from the <code>error_details</code> object, which includes the violation code in case the license is valid but its terms are violated.
repair_required	Cluster is in degraded mode and can only accept replacement nodes. When this happens, <code>error_details</code> contains two fields: <code>failed_nodes</code> and <code>replace_candidate</code> . The <code>failed_nodes</code> field is an array of objects, each describing a failed node with at least a <code>uid</code> field and an optional <code>rack_id</code> . <code>replace_candidate</code> is the UID of the node most suitable for replacement.
insufficient_node_memory	An attempt to replace a dead node fails because the replaced node does not have enough memory. When this happens, <code>error_details</code> contains a <code>required_memory</code> field which indicates the node memory requirement.
insufficient_node_flash	An attempt to replace a dead node fails because the replaced node does not have enough flash. When this happens, <code>error_details</code> contains a <code>required_flash</code> field which indicates the node flash requirement.
time_not_sync	An attempt to join a node with system time not synchronized with the rest of the cluster.
rack_id_required	An attempt to join a node with no <code>rack_id</code> in a rack-aware cluster. In addition, a <code>current_rack_ids</code> field will include an array of currently used rack ids.
socket_directory_mismatch	An attempt to join a node with a socket directory setting that differs from the cluster
node_config_mismatch	An attempt to join a node with a configuration setting (e.g. <code>confdir</code> , <code>osuser</code> , <code>installdir</code>) that differs from the cluster
path_error	A needed path does not exist or is not accessible.
internal_error	A different, unspecified internal error was encountered.

Status codes

Code	Description
200 OK	No error

Start bootstrapping

```
POST /v1/bootstrap/{action}
```

Initiate bootstrapping.

The request must contain a bootstrap configuration JSON object, as described in [Object attributes](#) or a minimal subset.

Bootstrapping is permitted only when the current bootstrap state is `idle` or `error` (in which case the process will restart with the new configuration).

This request is asynchronous - once the request has been accepted, the caller is expected to poll bootstrap status while waiting for it to complete.

Request

Example HTTP request

```
POST /bootstrap/create_cluster
```

Example JSON body

Join cluster

```
{
  "action": "join_cluster",
  "cluster": {
    "nodes": [ "1.1.1.1", "2.2.2.2" ]
  },
  "node": {
    "paths": {
      "persistent_path": "/path/to/persistent/storage",
      "ephemeral_path": "/path/to/ephemeral/storage",
      "bigstore_path": "/path/to/bigstore/storage"
    },
    "bigstore_driver": "speedb",
    "identity": {
      "addr": "1.2.3.4",
      "external_addr": ["2001:0db8:85a3:0000:0000:8a2e:0370:7334", "3.4.5.6"]
    }
  },
  "credentials": {
    "username": "my_username",
    "password": "my_password"
  }
}
```

Create cluster

```
{
  "action": "create_cluster",
  "cluster": {
    "nodes": [],
    "name": "my.cluster"
  },
  "node": {
    "paths": {
      "persistent_path": "/path/to/persistent/storage",
      "ephemeral_path": "/path/to/ephemeral/storage",
      "bigstore_path": "/path/to/bigredis/storage"
    },
    "identity": {
      "addr": "1.2.3.4",
      "external_addr": ["2001:0db8:85a3:0000:0000:8a2e:0370:7334", "3.4.5.6"]
    },
    "bigstore_driver": "rocksdb"
  },
  "license": "",
  "credentials": {
    "username": "my_username",
    "password": "my_password"
  }
}
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [bootstrap object](#) in the request body.

Response

Status codes

Code	Description
200 OK	Request received and processing begins.
409 Conflict	Bootstrap already in progress (check state)

Updated: August 17, 2023

Bootstrap validation requests

Method	Path	Description
POST	/v1/bootstrap/validate/{action}	Perform bootstrap validation

Bootstrap validation

POST /v1/bootstrap/validate/{action}

Perform bootstrap validation.

Unlike actual bootstrapping, this request blocks and immediately returns with a response.

Request

Example HTTP request

```
POST /bootstrap/validate/join_cluster
```

Request body

The request must contain a [bootstrap configuration object](#), similar to the one used for actual bootstrapping.

Response

If an error occurs, the call returns a `bootstrap_status` JSON object that contains the following fields:

Field	Description
	Current bootstrap state. idle: No bootstrapping started. initiated: Bootstrap request received. creating_cluster: In the process of creating a new cluster. joining_cluster: In the process of joining an existing cluster. error: The last bootstrap action failed. completed: The last bootstrap action completed successfully.
state	Bootstrap process start time
start_time	Bootstrap process end time
end_time	
error_code	If state is <code>error</code> , this error code describes the type of error encountered.

Field	Description
error_details	An error-specific object that may contain additional information about the error. A common field in use is <code>message</code> which provides a more verbose error message.

Status codes

Code	Description
200 OK	No error, validation was successful.
406 Not Acceptable	Validation failed, bootstrap status is returned as body.

Updated: November 29, 2021

Cluster requests

Method	Path	Description
GET	/v1/cluster	Get cluster info
PUT	/v1/cluster	Update cluster settings

Get cluster info

GET /v1/cluster

Get cluster info.

Required permissions

Permission name

[view_cluster_info](#)

Request

Example HTTP request

GET /cluster

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a [cluster object](#).

Example JSON body

```
{
  "name": "my-rlec-cluster",
  "alert_settings": { "..."}, 
  "created_time": "2015-04-29T09:09:25Z",
  "email_alerts": false,
  "email_from": "",
  "rack_aware": false,
  "smtp_host": "",
  "smtp_password": "",
  "smtp_port": 25,
  "smtp_tls_mode": "none",
  "smtp_username": ""
}
```

Status codes

Code	Description
200 OK	No error

Update cluster settings

PUT /v1/cluster

Update cluster settings.

If called with the `dry_run` URL query string, the function will validate the [cluster object](#), but will not apply the requested changes.

Required permissions

Permission name

[update_cluster](#)

Request

Example HTTP request

PUT /cluster

Example JSON body

```
{
  "email_alerts": true,
  "alert_settings": {
    "node_failed": true,
    "node_memory": {
      "enabled": true,
      "threshold": "80"
    }
  }
}
```

The above request will enable email alerts and alert reporting for node failures and node removals.

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name

Key	Value	Description
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
dry_run	string	Validate but don't apply the new cluster settings

Request body

Include a [cluster object](#) with updated fields in the request body.

Response

Example JSON body

```
{
  "name": "mycluster.mydomain.com",
  "email_alerts": true,
  "alert_settings": {
    "node_failed": true,
    "node_memory": {
      "enabled": true,
      "threshold": "80"
    }
  },
  // additional fields...
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
bad_nginx_conf	<ul style="list-style-type: none"> Designated port is already bound. nginx configuration is illegal. Debuginfo path doesn't exist.
bad_debuginfo_path	<ul style="list-style-type: none"> Debuginfo path is inaccessible.
config_edit_conflict	Cluster config was edited by another source simultaneously.

Status codes

Code	Description
200 OK	No error.
400 Bad Request	Bad content provided.

Updated: November 29, 2021

Cluster actions requests

Method	Path	Description
--------	------	-------------

Method	Path	Description
GET	/v1/cluster/actions	Get the status of all actions
GET	/v1/cluster/actions/{action}	Get the status of a specific action
POST	/v1/cluster/actions/{action}	Initiate a cluster-wide action
DELETE	/v1/cluster/actions/{action}	Cancel action or remove action status

Get all cluster actions

GET /v1/cluster/actions

Get the status of all currently executing, queued, or completed cluster actions.

Required permissions

Permission name

[view_status_of_cluster_action](#)

Request

Example HTTP request

GET /cluster/actions

Response

Returns a JSON array of [action objects](#).

Example JSON body

```
{
  "actions": [
    {
      "name": "action_name",
      "status": "queued",
      "progress": 0.0
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error, response provides info about an ongoing action.
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run).

Get cluster action

GET /v1/cluster/actions/{action}

Get the status of a currently executing, queued, or completed cluster action.

Required permissions

Permission name

Permission name

[view_status_of_cluster_action](#)

Request

Example HTTP request

```
GET /cluster/actions/action_name
```

URL parameters

Field	Type	Description
action	string	The action to check.

Response

Returns an [action object](#).

Example JSON body

```
{
  "name": "action_name",
  "status": "queued",
  "progress": 0.0
}
```

Status codes

Code	Description
200 OK	No error, response provides info about an ongoing action.
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run).

Initiate cluster-wide action

```
POST /v1/cluster/actions/{action}
```

Initiate a cluster-wide action.

The API allows only a single instance of any action type to be invoked at the same time, and violations of this requirement will result in a [409 CONFLICT](#) response.

The caller is expected to query and process the results of the previously executed instance of the same action, which will be removed as soon as the new one is submitted.

Required permissions

Permission name

[start_cluster_action](#)

Request

Example HTTP request

```
POST /cluster/actions/action_name
```

URL parameters

Field	Type	Description
action	string	The name of the action required.

Response

The body content may provide additional action details. Currently, it is not used.

Status codes

Code	Description
200 OK	No error, action was initiated.
400 Bad Request	Bad action or content provided.
409 Conflict	A conflicting action is already in progress.

Cancel action

```
DELETE /v1/cluster/actions/{action}
```

Cancel a queued or executing cluster action, or remove the status of a previously executed and completed action.

Required permissions

Permission name

[cancel_cluster_action](#)

Request

Example HTTP request

```
DELETE /v1/cluster/actions/action_name
```

URL parameters

Field	Type	Description
action	string	The name of the action to cancel, currently no actions are supported.

Response

Returns a status code.

Status codes

Code	Description
200 OK	Action will be cancelled when possible.
404 Not Found	Action unknown or not currently running.

Updated: November 29, 2021

Cluster alerts requests

Method	Path	Description
GET	/v1/cluster/alerts	Get all cluster alerts
GET	/v1/cluster/alerts/{alert}	Get a specific cluster alert

Get all cluster alerts

GET /v1/cluster/alerts

Get all alert states for the cluster object.

Required permissions

Permission name

view_cluster_alerts

Request

Example HTTP request

GET /cluster/alerts

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
ignore_settings	boolean	Retrieve updated alert state regardless of the cluster's alert_settings. When not present, a disabled alert will always be retrieved as disabled with a false state. (optional)

Response

Returns a hash of [alert objects](#) and their states.

Example JSON body

```
{
  "cluster_too_few_nodes_for_replication": {
    "change_time": "2014-12-22T11:48:00Z",
    "change_value": {
      "state": false
    },
    "enabled": true,
    "state": "off",
    "severity": "WARNING",
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error

Get cluster alert

```
GET /v1/cluster/alerts/{alert}
```

Get a cluster alert state.

Required permissions

Permission name

[view_cluster_alerts](#)

Request

Example HTTP request

```
GET /cluster/alerts/cluster_too_few_nodes_for_replication
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
ignore_settings	boolean	Retrieve updated alert state regardless of the cluster's alert_settings. When not present, a disabled alert will always be retrieved as disabled with a false state. (optional)

Response

Returns an [alert object](#).

Example JSON body

```
{
  "change_time": "2014-12-22T11:48:00Z",
  "change_value": {
    "state": false
  },
  "enabled": true,
  "state": "off",
  "severity": "WARNING",
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified alert does not exist

Auditing database connections requests

Method	Path	Description
GET	/v1/cluster/auditing/db_conns	Get database connection auditing settings
PUT	/v1/cluster/auditing/db_conns	Update database connection auditing settings
DELETE	/v1/cluster/auditing/db_conns	Delete database connection auditing settings

Get database auditing settings

```
GET /v1/cluster/auditing/db_conns
```

Gets the configuration settings for [auditing database connections](#).

Required permissions

Permission name

[view_cluster_info](#)

Request

Example HTTP request

```
GET /cluster/auditing/db_conns
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a [database connection auditing configuration object](#).

Example JSON body

```
{
  "audit_address": "127.0.0.1",
  "audit_port": 12345,
  "audit_protocol": "TCP",
  "audit_reconnect_interval": 1,
  "audit_reconnect_max_attempts": 0
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
<code>db_conns_auditing_unsupported_by_capability</code>	Not all nodes support DB Connections Auditing capability

Status codes

Code	Description
200 OK	Success
406 Not Acceptable	Feature not supported for all nodes

Update database auditing

PUT /v1/cluster/auditing/db_conn

Updates the configuration settings for [auditing database connections](#).

Required permissions

Permission name

update_cluster

Request

Example HTTP request

PUT /cluster/auditing/db_conn

Example JSON body

```
{  
    "audit_protocol": "TCP",  
    "audit_address": "127.0.0.1",  
    "audit_port": 12345,  
    "audit_reconnect_interval": 1,  
    "audit_reconnect_max_attempts": 0  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [database connection auditing configuration object](#) with updated fields in the request body.

Response

Returns the updated [database connection auditing configuration object](#).

Example JSON body

```
{
  "audit_address": "127.0.0.1",
  "audit_port": 12345,
  "audit_protocol": "TCP",
  "audit_reconnect_interval": 1,
  "audit_reconnect_max_attempts": 0
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
db_conns_auditing_unsupported_by_capability	Not all nodes support DB Connections Auditing capability

Status codes

Code	Description
200 OK	Success
406 Not Acceptable	Feature not supported for all nodes

Delete database auditing settings

`DELETE /v1/cluster/auditing/db_conns`

Resets the configuration settings for [auditing database connections](#).

Required permissions

Permission name
update_cluster

Request

Example HTTP request

`DELETE /cluster/auditing/db_conns`

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a status code that indicates whether the database connection auditing settings reset successfully or failed to reset.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
------	-------------

Code	Description
db_conns_audit_config_not_found	Unable to find the auditing configuration
cannot_delete_audit_config_when_policy_enabled	Auditing cluster policy is 'enabled' when trying to delete the auditing configuration
cannot_delete_audit_config_when_bdb_enabled	One of the databases has auditing configuration 'enabled' when trying to delete the auditing configuration

Status codes

Code	Description
200 OK	Success
404 Not Found	Configuration not found
406 Not Acceptable	Feature not supported for all nodes

Updated: February 9, 2023

Cluster certificates requests

Method	Path	Description
GET	/v1/cluster/certificates	Get cluster certificates
DELETE	/v1/cluster/certificates/{certificate_id}	Delete cluster certificate

Get cluster certificates

GET /v1/cluster/certificates

Get the cluster's certificates.

Required permissions

Permission name

[view_cluster_info](#)

Request

Example HTTP request

GET /cluster/certificates

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON object that contains the cluster's certificates and keys.

Example JSON body

```
{
  "api_cert": "-----BEGIN CERTIFICATE-----...-----END CERTIFICATE-----",
  "api_key": "-----BEGIN RSA PRIVATE KEY-----...-----END RSA PRIVATE KEY-----"
  "// additional certificates..."
}
```

Status codes

Code	Description
200 OK	No error

Delete cluster certificate

`DELETE /v1/cluster/certificates/{string: certificate_name}`

Removes the specified cluster certificate from both CCS and disk across all nodes. Only optional certificates can be deleted through this endpoint. See the [certificates table](#) for the list of cluster certificates and their descriptions.

Request

Example HTTP request

`DELETE /cluster/certificates/<certificate_name>`

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a status code that indicates the certificate deletion success or failure.

Status codes

Code	Description
200 OK	Operation successful
404 Not Found	Failed, requested deletion of an unknown certificate
403 Forbidden	Failed, requested deletion of a required certificate
500 Internal Server Error	Failed, error while deleting certificate from disk

Updated: December 23, 2022

Rotate cluster certificates requests

Method	Path	Description
POST	<code>/v1/cluster/certificates/rotate</code>	Regenerate all internal cluster certificates

Rotate cluster certificates

```
POST /v1/cluster/certificates/rotate
```

Regenerates all *internal* cluster certificates.

The certificate rotation will be performed on all nodes within the cluster. If “name” is provided, only rotate the specified certificate on all nodes within the cluster.

Request

Example HTTP request

```
POST /cluster/certificates/rotate
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Responds with a 200 OK status code if the internal certificates successfully rotate across the entire cluster.

Status codes

Code	Description
200 OK	No error
400 Bad Request	Failed, not all nodes have been updated.
403 Forbidden	Unsupported internal certificate rotation.

Updated: November 29, 2021

Cluster LDAP requests

Method	Path	Description
GET	/v1/cluster/ldap	Get LDAP configuration
PUT	/v1/cluster/ldap	Set/update LDAP configuration
DELETE	/v1/cluster/ldap	Delete LDAP configuration

Get LDAP configuration

```
GET /v1/cluster/ldap
```

Get the LDAP configuration.

Required permissions

Permission name

[view_ldap_config](#)

Request

Example HTTP request

```
GET /cluster/ldap
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns an [LDAP object](#).

Example JSON body

```
{
  "bind_dn": "rl_admin",
  "bind_pass": "***",
  "ca_cert": "",
  "control_plane": false,
  "data_plane": false,
  "dn_group_attr": "MemberOf",
  "dn_group_query": {},
  "starttls": false,
  "uris": ["ldap://ldap.example.org:636"],
  "user_dn_query": {},
  "user_dn_template": "cn=%u, ou=users,dc=example,dc=org"
}
```

Status codes

Code	Description
200 OK	Success

Update LDAP configuration

```
PUT /v1/cluster/ldap
```

Set or update the cluster LDAP configuration.

Required permissions

Permission name

[config_ldap](#)

Request

Example HTTP request

```
POST /cluster/ldap
```

Example JSON body

```
{
  "uris": [
    "ldap://ldap.redislabs.com:389"
  ],
  "bind_dn": "rl_admin",
  "bind_pass": "secret",
  "user_dn_template": "cn=%u,dc=example,dc=org",
  "dn_group_attr": "MemberOf"
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include an [LDAP object](#) with updated fields in the request body.

Response

Returns a status code. If an error occurs, the response body may include an error code and message with more details.

Error codes

Possible error_code values:

Code	Description
illegal_fields_combination	An unacceptable combination of fields was specified for the configuration object (e.g.: two mutually-exclusive fields), or a required field is missing.

Status codes

Code	Description
200 OK	Success, LDAP config has been set.
400 Bad Request	Bad or missing configuration parameters.

Delete LDAP configuration

```
DELETE /v1/cluster/ldap
```

Clear the LDAP configuration.

Required permissions

Permission name
config_ldap

Request

Example HTTP request

```
DELETE /cluster/ldap
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a status code.

Status codes

Code	Description
200 OK	Success

Updated: March 3, 2022

Cluster module capabilities requests

Method	Path	Description
GET	/v1/cluster/module-capabilities	List possible Redis module capabilities

List Redis module capabilities

`GET /v1/cluster/module-capabilities`

List possible Redis module capabilities.

Required permissions

Permission name

[view_cluster_modules](#)

Request

Example HTTP request

`GET /cluster/module-capabilities`

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	*/*	Accepted media type

Response

Returns a JSON object that contains a list of capability names and descriptions.

Example JSON body

```
{
  "all_capabilities": [
    {"name": "types", "desc": "module has its own types and not only operate on existing redis types"},  

    {"name": "no_multi_key", "desc": "module has no methods that operate on multiple keys"}  

    "// additional capabilities..."
  ]
}
```

Status codes

Code	Description
200 OK	No error

Updated: November 29, 2021

Cluster policy requests

Method	Path	Description
GET	/v1/cluster/policy	Get cluster policy settings
PUT	/v1/cluster/policy	Update cluster policy settings

Get cluster policy

GET /v1/cluster/policy

Gets the cluster's current policy settings.

Required permissions

Permission name
view_cluster_info

Request

Example HTTP request

GET /cluster/policy

Request headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a [cluster settings object](#).

Example JSON body

```
{
  "db_conns_auditing": false,
  "default_non_sharded_proxy_policy": "single",
  "default_provisioned_redis_version": "6.0",
  "default_sharded_proxy_policy": "single",
  "default_shards_placement": "dense",
  "redis_upgrade_policy": "major",
  "// additional fields..."
}
```

Status codes

Code	Description
200 OK	Success

Update cluster policy

PUT /v1/cluster/policy

Update cluster policy settings.

Required permissions

Permission name
update_cluster

Request

Example HTTP request

PUT /cluster/policy

Example JSON body

```
{
  "default_shards_placement": "sparse",
  "default_sharded_proxy_policy": "all-nodes"
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [cluster settings object](#) with updated fields in the request body.

Response

Returns a status code that indicates the success or failure of the cluster settings update.

Status codes

Code	Description
200 OK	Success

Code	Description
200 OK	Success
400 Bad Request	Failed to set parameters

Updated: February 9, 2023

Cluster services configuration requests

Method	Path	Description
GET	/v1/cluster/services_configuration	Get cluster services settings
PUT	/v1/cluster/services_configuration	Update cluster services settings

Get cluster services configuration

GET /v1/cluster/services_configuration

Get cluster services settings.

Required permissions

Permission name

view_cluster_info

Request

Example HTTP request

```
GET /cluster/services_configuration
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a [services configuration object](#).

Example JSON body

```
{
    "cm_server": {
        "operating_mode": "disabled"
    },
    "mdns_server": {
        "operating_mode": "enabled"
    },
    "// additional services..."
}
```

Status codes

Code	Description
200 OK	No error

Update cluster services configuration

PUT /v1/cluster/services_configuration

Update the cluster services settings.

Required permissions

Permission name

update_cluster

Request

Example HTTP request

PUT /cluster/services_configuration

Example JSON body

```
{  
    "cm_server": {  
        "operating_mode": "disabled"  
    },  
    "// additional services..."  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [services configuration object](#) with updated fields in the request body.

Response

Returns the updated [services configuration object](#).

Example JSON body

```
{  
    "cm_server": {  
        "operating_mode": "disabled"  
    },  
    "mdns_server": {  
        "operating_mode": "enabled"  
    },  
    "// additional services..."  
}
```

Status codes

Code	Description
200 OK	No error

Updated: March 18, 2022

Cluster stats requests

Method	Path	Description
GET	/v1/cluster/stats	Get cluster stats

Get cluster stats

```
GET /v1/cluster/stats
```

Get cluster statistics.

Permissions

Permission name	Roles
view_cluster_stats	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /cluster/stats/1?interval=1hour&stime=2014-08-28T10:00:00Z
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for the cluster.

Example JSON body

```
{  
  "intervals": [  
    {  
      "interval": "1hour",  
      "stime": "2015-05-27T12:00:00Z",  
      "etime": "2015-05-28T12:59:59Z",  
      "conns": 0.0,  
      "cpu_idle": 0.8533959401503577,  
      "cpu_system": 0.01602159448549579,  
      "cpu_user": 0.08721123782294203,  
      "egress_bytes": 1111.2184745131947,  
      "ephemeral_storage_avail": 3406676307.1449075,  
      "ephemeral_storage_free": 4455091440.360014,  
      "free_memory": 2745470765.673594,  
      "ingress_bytes": 220.84083194769272,  
      "interval": "1week",  
      "persistent_storage_avail": 3406676307.1533995,  
      "persistent_storage_free": 4455091440.088265,  
      "total_req": 0.0  
    },  
    {  
      "interval": "1hour",  
      "stime": "2015-05-27T13:00:00Z",  
      "etime": "2015-05-28T13:59:59Z",  
      "// additional fields..."  
    }  
  ]  
}
```

Example requests

cURL

```
$ curl -k -u "[username]:[password]" -X GET  
  https://[host][:port]/v1/cluster/stats?interval=1hour
```

Python

```
import requests  
  
url = "https://[host][:port]/v1/cluster/stats?interval=1hour"  
auth = ("[username]", "[password]")  
  
response = requests.request("GET", url, auth=auth)  
  
print(response.text)
```

Status codes

Code	Description
200 OK	No error
500 Internal Server Error	Internal server error

Updated: August 18, 2022

Cluster last stats requests

Method	Path	Description
GET	/v1/cluster/stats/last	Get most recent cluster stats

Get latest cluster stats

GET /v1/cluster/stats/last

Get the most recent cluster statistics.

Required permissions

Permission name

[view_cluster_stats](#)

Request

Example HTTP request

```
GET /cluster/stats/last?interval=1sec&stime=2015-10-14T06:44:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week. Default: 1sec. (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns the most recent [statistics](#) for the cluster.

Example JSON body

```
{
  "conns": 0.0,
  "cpu_idle": 0.842499999988358,
  "cpu_system": 0.0174999999992724,
  "cpu_user": 0.0837499999978172,
  "egress_bytes": 7403.0,
  "ephemeral_storage_avail": 151638712320.0,
  "ephemeral_storage_free": 162375925760.0,
  "etime": "2015-10-14T06:44:01Z",
  "free_memory": 5862400000.0,
  "ingress_bytes": 7469.0,
  "interval": "1sec",
  "persistent_storage_avail": 151638712320.0,
  "persistent_storage_free": 162375925760.0,
  "stime": "2015-10-14T06:44:00Z",
  "total_req": 0.0
}
```

Status codes

Code	Description
200 OK	No error
500 Internal Server Error	Internal server error

Updated: November 29, 2021

Update cluster certificate requests

Method	Path	Description
PUT	/v1/cluster/update_cert	Update a cluster certificate

Update cluster certificate

```
PUT /v1/cluster/update_cert
```

Replaces an existing certificate on all nodes within the cluster with a new certificate. The new certificate must pass validation before it can replace the old certificate.

See the [certificates table](#) for the list of cluster certificates and their descriptions.

Request

Example HTTP request

```
PUT /cluster/update_cert
```

Example JSON body

```
{
  "name": "certificate1",
  "key": "-----BEGIN RSA PRIVATE KEY-----\n[key_content]\n-----END RSA PRIVATE KEY-----",
  "certificate": "-----BEGIN CERTIFICATE-----\n[cert_content]\n-----END CERTIFICATE-----"
}
```

Replace [key_content] with the content of the private key and [cert_content] with the content of the certificate.

Response

Responds with the 200 OK status code if the certificate replacement succeeds across the entire cluster.

Otherwise, retry the certificate update in case the failure was due to a temporary issue in the cluster.

Status codes

Code	Description
200 OK	No error
400 Bad Request	Failed, invalid certificate.
403 Forbidden	Failed, unknown certificate.
404 Not Found	Failed, invalid certificate.
406 Not Acceptable	Failed, expired certificate.
409 Conflict	Failed, not all nodes have been updated.

Updated: December 23, 2022

CRDB tasks requests

Method	Path	Description
GET	/v1/crdb_tasks/{task_id}	Get the status of an executed task

Get task status

GET /v1/crdb_tasks/{task_id}

Get the status of an executed task.

The status of a completed task is kept for 500 seconds by default.

Request

Example HTTP request

```
GET /crdb_tasks/1
```

Request headers

Key	Value	Description
X-Result-TTL	integer	Task time to live

URL parameters

Field	Type	Description
task_id	string	Task ID

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	Task status.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Task not found.

Updated: November 29, 2021

CRDBs requests

Method	Path	Description
GET	/v1/crdbs	Get all Active-Active databases
GET	/v1/crdbs/{crdb_guid}	Get a specific Active-Active database
PATCH	/v1/crdbs/{crdb_guid}	Update an Active-Active database
POST	/v1/crdbs	Create a new Active-Active database
DELETE	/v1/crdbs/{crdb_guid}	Delete an Active-Active database

Get all Active-Active databases

```
GET /v1/crdbs
```

Get a list of all Active-Active databases on the cluster.

Request

Example HTTP request

```
GET /crdbs
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

Response

Returns a JSON array of [CRDB objects](#).

Status codes

Code	Description
200 OK	A list of Active-Active database.
401 Unauthorized	Unauthorized request. Invalid credentials

Get an Active-Active database

```
GET /v1/crdbs/{crdb_guid}
```

Get a specific Active-Active database.

Request

Example HTTP request

```
GET /crdbs/552bbccb-99f3-4142-bd17-93d245f0bc79
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Query parameters

Field	Type	Description
instance_id	integer	Instance from which to get the Active-Active database information

Response

Returns a [CRDB object](#).

Status codes

Code	Description
200 OK	Active-Active database information is returned.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Database or configuration does not exist.

Update an Active-Active database

```
PATCH /v1/crdbs/{crdb_guid}
```

Update an Active-Active database's configuration.

In order to add or remove instances, use [POST crdbs/{crdb_guid}/updates](#) instead.

Request

Example HTTP request

```
PATCH /crdbs/552bbccb-99f3-4142-bd17-93d245f0bc79
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Request body

Include a [CRDB object](#) with updated fields in the request body.

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	The request has been accepted.
400 Bad Request	The posted Active-Active database contains invalid parameters.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Configuration or Active-Active database not found.
406 Not Acceptable	The posted Active-Active database cannot be accepted.

Create an Active-Active database

```
POST /v1/crdbs
```

Create a new Active-Active database.

Request

Example HTTP request

```
POST /crdbs
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

Request body

Include a [CRDB object](#), which defines the Active-Active database, in the request body.

Example body

```

{
  "default_db_config": [
    {
      "name": "sample-crdb",
      "memory_size": 214748365
    },
    "instances": [
      [
        {
          "cluster": [
            {
              "url": "http://<cluster1_FQDN>:9443",
              "credentials": [
                {
                  "username": "<username>",
                  "password": "<password>"
                },
                "name": "cluster-1"
              ],
              "compression": 6
            },
            {
              "cluster": [
                {
                  "url": "http://<cluster2_FQDN>:9443",
                  "credentials": [
                    {
                      "username": "<username>",
                      "password": "<password>"
                    },
                    "name": "cluster-2"
                  ],
                  "compression": 6
                }
              ],
              "name": "sample-crdb"
            }
          ]
        }
      ]
    }
  ]
}

```

This JSON body creates an Active-Active database without TLS and with two participating clusters.

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	The request has been accepted.
400 Bad Request	The request is invalid or malformed.
401 Unauthorized	Unauthorized request. Invalid credentials
406 Not Acceptable	The posted Active-Active database cannot be accepted.

Delete an Active-Active database

```
DELETE /v1/crdbs/{crdb_guid}
```

Delete an Active-Active database.

Request

Example HTTP request

```
DELETE /crdbs/552bbccb-99f3-4142-bd17-93d245f0bc79
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	Action was successful.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Configuration or Active-Active database not found.
406 Not Acceptable	The Active-Active GUID is invalid or the Active-Active database was already deleted.

Updated: August 3, 2022

CRDB flush requests

Method	Path	Description
PUT	/v1/crdbs/{crdb_guid}/flush	Flush an Active-Active database

Flush an Active-Active database

```
PUT /v1/crdbs/{crdb_guid}/flush
```

Flush an Active-Active database.

Request

Example HTTP request

```
PUT /crdbs/552bbccb-99f3-4142-bd17-93d245f0bc79/flush
```

Headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

Key	Value	Description
-----	-------	-------------

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	Action was successful.
400 Bad Request	The request is invalid or malformed.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Configuration or Active-Active database not found.
406 Not Acceptable	Configuration cannot be accepted, typically because it was already committed.

Updated: August 3, 2022

CRDB health report requests

Method	Path	Description
GET	/v1/crdbs/{crdb_guid}/health_report	Get a health report for an Active-Active database

Get health report

```
GET /v1/crdbs/{crdb_guid}/health_report
```

Get a health report for an Active-Active database.

Request

Example HTTP request

```
GET /crdbs/{crdb_guid}/health_report
```

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Query parameters

Field	Type	Description
-------	------	-------------

Field	Type	Description
instance_id	integer	The request retrieves information from the specified Active-Active database instance. If this instance doesn't exist, the request retrieves information from all instances. (optional)

Response

Returns a JSON array of [CRDB health report objects](#).

Status codes

Code	Description
200 OK	Action was successful.
404 Not Found	Configuration or Active-Active database not found.

Updated: April 13, 2022

CRDB purge requests

Method	Path	Description
PUT	/v1/crdbs/{crdb_guid}/purge	Purge data from an instance that was forcibly removed from the Active-Active database

Purge data from removed instance

`PUT /v1/crdbs/{crdb_guid}/purge`

Purge the data from an instance that was removed from the Active-Active database by force.

When you force the removal of an instance from an Active-Active database, the removed instance keeps the data and configuration according to the last successful synchronization.

To delete the data and configuration from the forcefully removed instance you must use this API (Must be executed locally on the removed instance).

Request

Example HTTP request

`PUT /crdbs/1/purge`

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Request body

Field	Type	Description
instances	array of integers	Array of unique instance IDs

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	Action was successful.
400 Bad Request	The request is invalid or malformed.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Configuration, instance, or Active-Active database not found.

Updated: November 29, 2021

CRDB updates requests

Method	Path	Description
POST	/v1/crdbs/{crdb_guid}/updates	Modify Active-Active configuration

Modify Active-Active configuration

POST /v1/crdbs/{crdb_guid}/updates

Modify Active-Active configuration.



Warning - This is a very powerful API request and can cause damage if used incorrectly.

In order to add or remove instances, you must use this API. For simple configuration updates, it is recommended to use PATCH on /crdbs/{crdb_guid} instead.

Updating default_db_config affects both existing and new instances that may be added.

When you update db_config, it changes the configuration of the database that you specify. This field overrides corresponding fields (if any) in default_db_config.

Request

Example HTTP request

POST /crdbs/1/updates

Request headers

Key	Value	Description
X-Task-ID	string	Specified task ID
X-Result-TTL	integer	Time (in seconds) to keep task result

URL parameters

Field	Type	Description
crdb_guid	string	Globally unique Active-Active database ID (GUID)

Request body

Include a [CRDB modify_request object](#) with updated fields in the request body.

Response

Returns a [CRDB task object](#).

Status codes

Code	Description
200 OK	The request has been accepted.
400 Bad Request	The posted Active-Active database contains invalid parameters.
401 Unauthorized	Unauthorized request. Invalid credentials
404 Not Found	Configuration, instance or Active-Active database not found.
406 Not Acceptable	The posted Active-Active database cannot be accepted.

Updated: April 13, 2022

Endpoints stats requests

Method	Path	Description
GET	/v1/endpoints/stats	Get stats for all endpoints

Get all endpoints stats

`GET /v1/endpoints/stats`

Get statistics for all endpoint-proxy links.

 Note: This method will return both endpoints and listeners stats for backwards compatibility.

Required permissions

Permission name

[view_endpoint_stats](#)

Request

Example HTTP request

```
GET /endpoints/stats?interval=1hour&stime=2014-08-28T10:00:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)

Field	Type	Description
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

The uid format in the response is: "BDB_UID:ENDPOINT_UID:PROXY_UID"

For example: { "uid": "1:2:3" } means BDB_UID=1,ENDPOINT_UID=2, and PROXY_UID=3

Example JSON body

```
[
  {
    "uid" : "365:1:1",
    "intervals" : [
      {
        "interval" : "10sec",
        "total_req" : 0,
        "egress_bytes" : 0,
        "cmd_get" : 0,
        "monitor_sessions_count" : 0,
        "auth_errors" : 0,
        "acc_latency" : 0,
        "stime" : "2017-01-12T14:59:50Z",
        "write_res" : 0,
        "total_connections_received" : 0,
        "cmd_set" : 0,
        "read_req" : 0,
        "max_connections_exceeded" : 0,
        "acc_write_latency" : 0,
        "write_req" : 0,
        "other_res" : 0,
        "cmd_flush" : 0,
        "acc_read_latency" : 0,
        "other_req" : 0,
        "conns" : 0,
        "write_started_res" : 0,
        "cmd_touch" : 0,
        "read_res" : 0,
        "auth_cmds" : 0,
        "etime" : "2017-01-12T15:00:00Z",
        "total_started_res" : 0,
        "ingress_bytes" : 0,
        "last_res_time" : 0,
        "read_started_res" : 0,
        "acc_other_latency" : 0,
        "total_res" : 0,
        "last_req_time" : 0,
        "other_started_res" : 0
      }
    ],
    },
    {
      "intervals" : [
        {
          "acc_read_latency" : 0,
          "other_req" : 0,
          "etime" : "2017-01-12T15:00:00Z",
          "auth_cmds" : 0,
          "total_started_res" : 0,
          "....+ other stats" : 0
        }
      ]
    }
  }
]
```

```

    "write_started_res" : 0,
    "cmd_touch" : 0,
    "conns" : 0,
    "read_res" : 0,
    "total_res" : 0,
    "other_started_res" : 0,
    "last_req_time" : 0,
    "read_started_res" : 0,
    "last_res_time" : 0,
    "ingress_bytes" : 0,
    "acc_other_latency" : 0,
    "egress_bytes" : 0,
    "interval" : "10sec",
    "total_req" : 0,
    "acc_latency" : 0,
    "auth_errors" : 0,
    "cmd_get" : 0,
    "monitor_sessions_count" : 0,
    "read_req" : 0,
    "max_connections_exceeded" : 0,
    "total_connections_received" : 0,
    "cmd_set" : 0,
    "acc_write_latency" : 0,
    "write_req" : 0,
    "stime" : "2017-01-12T14:59:50Z",
    "write_res" : 0,
    "cmd_flush" : 0,
    "other_res" : 0
  }
],
"uid" : "333:1:2"
}
]

```

Status codes

Code	Description
200 OK	No error

Updated: November 29, 2021

JSON schema requests

Method	Path	Description
GET	/v1/jsonschema	Get JSON schema of API objects

Get object JSON schema

GET /v1/jsonschema

Get the JSON schema of various [Redis Enterprise REST API objects](#).

Request

Example HTTP request

GET /jsonschema?object=bdb

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
object	string	Optional. The API object name: 'cluster', 'node', 'bdb' etc.

Response

Returns the JSON schema of the specified API object.

Example JSON body

```
{  
    "type": "object",  
    "description": "An API object that represents a managed database in the cluster.",  
    "properties": {  
        "...."  
    },  
    "...."  
}
```

Status codes

Code	Description
200 OK	Success.
406 Not Acceptable	Invalid object.

Updated: November 29, 2021

LDAP mappings requests

Method	Path	Description
GET	/v1/ldap_mappings	Get all LDAP mappings
GET	/v1/ldap_mappings/{uid}	Get a single LDAP mapping
PUT	/v1/ldap_mappings/{uid}	Update an LDAP mapping
POST	/v1/ldap_mappings	Create a new LDAP mapping
DELETE	/v1/ldap_mappings/{uid}	Delete an LDAP mapping

Get all LDAP mappings

```
GET /v1/ldap_mappings
```

Get all LDAP mappings.

Required permissions

Permission name

[view_all_ldap_mappings_info](#)

Request

Example HTTP request

```
GET /ldap_mappings
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON array of [LDAP mapping objects](#).

Example JSON body

```
[  
  {  
    "uid": 17,  
    "name": "Admins",  
    "dn": "OU=ops.group,DC=redislabs,DC=com",  
    "email": "ops.group@redislabs.com",  
    "role_uids": ["1"],  
    "email_alerts": true,  
    "bdb_email_alerts": ["1", "2"],  
    "cluster_email_alerts": true  
  }  
]
```

Status codes

Code	Description
200 OK	No error

Get LDAP mapping

```
GET /v1/ldap_mappings/{int: uid}
```

Get a specific LDAP mapping.

Required permissions

Permission name

[view_ldap_mapping_info](#)

Request

Example HTTP request

```
GET /ldap_mappings/1
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The object's unique ID.

Response

Returns an [LDAP mapping object](#).

Example JSON body

```
{  
    "uid": 17,  
    "name": "Admins",  
    "dn": "OU=ops.group,DC=redislabs,DC=com",  
    "email": "ops.group@redislabs.com",  
    "role_uids": ["1"],  
    "email_alerts": true,  
    "bdb_email_alerts": ["1", "2"],  
    "cluster_email_alerts": true  
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
ldap_mapping_not_exist	An object does not exist

Status codes

Code	Description
200 OK	Success.
403 Forbidden	Operation is forbidden.
404 Not Found	ldap_mapping does not exist.
501 Not Implemented	Cluster doesn't support LDAP mappings yet.

Update LDAP mapping

```
PUT /v1/ldap_mappings/{int: uid}
```

Update an existing ldap_mapping object.

Required permissions

Permission name

Permission name

update_ldap_mapping

Request

Example HTTP request

```
PUT /ldap_mappings/17
```

Example JSON body

```
{  
  "dn": "OU=ops,DC=redislabs,DC=com",  
  "email": "ops@redislabs.com",  
  "email_alerts": true,  
  "bdb_email_alerts": ["1", "2"],  
  "cluster_email_alerts": true  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include an [LDAP mapping object](#) with updated fields in the request body.

Response

Example JSON body

```
{  
  "uid": 17,  
  "name": "Admins",  
  "dn": "OU=ops,DC=redislabs,DC=com",  
  "email": "ops@redislabs.com",  
  "role_uids": ["1"],  
  "email_alerts": true,  
  "bdb_email_alerts": ["1", "2"],  
  "cluster_email_alerts": true  
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists
ldap_mapping_not_exist	An object does not exist
invalid_dn_param	A dn parameter has an illegal value
invalid_name_param	A name parameter has an illegal value
invalid_role_uids_param	A role_uids parameter has an illegal value

Code	Description
invalid_account_id_param	An account_id parameter has an illegal value

Status codes

Code	Description
200 OK	Success, LDAP mapping is created.
400 Bad Request	Bad or missing configuration parameters.
404 Not Found	Attempting to change a non-existing LDAP mapping.
501 Not Implemented	Cluster doesn't support LDAP mapping yet.

Create LDAP mapping

POST /v1/ldap_mappings

Create a new LDAP mapping.

Required permissions

Permission name

create_ldap_mapping

Request

Example HTTP request

POST /ldap_mappings

Example JSON body

```
{  
    "name": "Admins",  
    "dn": "OU=ops.group,DC=redislabs,DC=com",  
    "email": "ops.group@redislabs.com",  
    "role_uids": ["1"]  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include an [LDAP mapping object](#) in the request body.

Response

Example JSON body

```
{
  "uid": 17,
  "name": "Admins",
  "dn": "OU=ops.group,DC=redislabs,DC=com",
  "email": "ops.group@redislabs.com",
  "role_uids": ["1"]
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists
missing_field	A needed field is missing
invalid_dn_param	A dn parameter has an illegal value
invalid_name_param	A name parameter has an illegal value
invalid_role_uids_param	A role_uids parameter has an illegal value

Status codes

Code	Description
200 OK	Success, an LDAP-mapping object is created.
400 Bad Request	Bad or missing configuration parameters.
501 Not Implemented	Cluster doesn't support LDAP mappings yet.

Delete LDAP mapping

```
DELETE /v1/ldap_mappings/{int: uid}
```

Delete an LDAP mapping object.

Required permissions

Permission name

[delete_ldap_mapping](#)

Request

Example HTTP request

```
DELETE /ldap_mappings/1
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The ldap_mapping unique ID.

Response

Returns a status code. If an error occurs, the response body may include a more specific error code and message.

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
ldap_mapping_not_exist	An object does not exist

Status codes

Code	Description
200 OK	Success, the ldap_mapping is deleted.
406 Not Acceptable	The request is not acceptable.
501 Not Implemented	Cluster doesn't support LDAP mappings yet.

Updated: November 29, 2021

License requests

Method	Path	Description
GET	/v1/license	Get license details
PUT	/v1/license	Update the license

Get license

GET /v1/license

Returns the license details, including license string, expiration, and supported features.

Required permissions

Permission name
view_license

Request

Example HTTP request

GET /license

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON object that contains the license details:

Name	Type/Value	Description
license	string	License data
cluster_name	string	The cluster name (FQDN)
expired	boolean	If the cluster key is expired (true or false)
activation_date	string	The date of the cluster key activation
expiration_date	string	The date of the cluster key expiration
key	string	License key
features	array of strings	Features supported by the cluster
owner	string	License owner
shards_limit	integer	The total number of shards allowed by the cluster key
ram_shards_limit	integer	The number of RAM shards allowed by the cluster key (as of v7.2)
flash_shards_limit	integer	The number of Flash shards (Auto Tiering) allowed by the cluster key (as of v7.2)

Example JSON body

```
{
  "license": "----- LICENSE START -----\\ndi+iK...KniI9\\n----- LICENSE END -----\\n",
  "expired": true,
  "activation_date": "2018-12-31T00:00:00Z",
  "expiration_date": "2019-12-31T00:00:00Z",
  "ram_shards_limit": 300,
  "flash_shards_limit": 100,
  "shards_limit": 400,
  "features": ["bigstore"],
  "owner": "Redis",
  "cluster_name": "mycluster.local",
  "key": "----- LICENSE START -----\\ndi+iK...KniI9\\n----- LICENSE END -----\\n"
}
```

Status codes

Code	Description
200 OK	License is returned in the response body.
404 Not Found	No license is installed.

Update license

PUT /v1/license

Validate and install a new license string.

Required permissions

Permission name

Permission name

[install_new_license](#)

Request

The request must be a JSON object with a single key named “license”.

Example HTTP request

```
PUT /license
```

Example JSON body

```
{  
    "license": "----- LICENSE START -----\\ndi+iK...KniI9\\n----- LICENSE END -----\\n"  
}
```

Request headers

Key	Value	Description
Accept	application/json	Accepted media type

Request body

Include a JSON object that contains the new license string in the request body.

Response

Returns an error if the new license is not valid.

Status codes

Code	Description
200 OK	License installed successfully.
400 Bad Request	Invalid request, either bad JSON object or corrupted license was supplied.
406 Not Acceptable	License violation. A response body provides more details on the specific cause.

Updated: August 17, 2023

Logs requests

Method	Path	Description
GET	/v1/logs	Get cluster events log

Get cluster events log

```
GET /v1/logs
```

Get cluster events log.

Required permissions

Permission name

[view_logged_events](#)

Request

Example HTTP request

```
GET /logs?order=desc
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
stime	ISO_8601	Start time before which we don't want events. (optional)
etime	ISO_8601	End time after which we don't want events. (optional)
order	string	desc/asc - get events in descending or ascending order. Defaults to asc.
limit	integer	Maximum number of events to return. (optional)
offset	integer	Skip offset events before returning first one (useful for pagination). (optional)

Response

Returns a JSON array of events.

Example JSON body

```
[  
  {  
    "time": "2014-08-29T11:19:49Z",  
    "type": "bdb_name_updated",  
    "severity": "INFO",  
    "bdb_uid": "1",  
    "old_val": "test",  
    "new_val": "test123"  
  },  
  {  
    "time": "2014-08-29T11:18:48Z",  
    "type": "cluster_bdb_created",  
    "severity": "INFO",  
    "bdb_uid": "1",  
    "bdb_name": "test"  
  },  
  {  
    "time": "2014-08-29T11:17:49Z",  
    "type": "cluster_node_joined",  
    "severity": "INFO",  
    "node_uid": 2  
  }  
]
```

Event object

Field	Description
time	Timestamp when event happened.
type	Event type. Additional fields may be available for certain event types.
additional fields	Additional fields may be present based on event type.

Status codes

Code	Description
200 OK	No error

Updated: November 29, 2021

Modules requests

Method	Path	Description
GET	/v1/modules	List available modules
GET	/v1/modules/{uid}	Get a specific module
POST	/v1/modules	Upload a new module (deprecated)
POST	/v2/modules	Upload a new module
DELETE	/v1/modules/{uid}	Delete a module (deprecated)
DELETE	/v2/modules/{uid}	Delete a module

List modules

```
GET /v1/modules
```

List available modules, i.e. modules stored within the CCS.

Permissions

Permission name	Roles
view_cluster_modules	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /modules
```

Headers

Key	Value	Description
Host	127.0.0.1:9443	Domain name
Accept	*/*	Accepted media type

Response

Returns a JSON array of [module objects](#).

Status codes

Code	Description
200 OK	No error

Get module

```
GET /v1/modules/{string: uid}
```

Get specific available modules, i.e. modules stored within the CCS.

Permissions

Permission name	Roles
view_cluster_modules	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /modules/1
```

Headers

Key	Value	Description
Host	127.0.0.1:9443	Domain name
Accept	*/*	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The module's unique ID.

Response

Returns a [module object](#).

Status codes

Code	Description
200 OK	No error
404 Not Found	Module does not exist.

Upload module v1

POST /v1/modules

 Note: POST /v1/modules is deprecated as of Redis Enterprise Software version 7.2. Use [POST /v2/modules](#) instead.

Uploads a new module to the cluster.

The request must contain a Redis module, bundled using [RedisModule Packer](#). For modules in Redis Stack, download the module from the [download center](#).

See [Install a module on a cluster](#) for more information.

Permissions

Permission name	Roles
update_cluster	admin

Request

Example HTTP request

```
POST /v1/modules
```

Headers

Key	Value	Description
Host	string	Domain name
Accept	*/*	Accepted media type
Content-Length	integer	Length of the request body in octets
Expect	100-continue	Requires particular server behaviors
Content-Type	multipart/form-data	Media type of request/response body

Response

Returns a status code. If an error occurs, the response body may include an error code and message with more details.

Error codes

The server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
no_module	Module wasn't provided or could not be found
invalid_module	Module either corrupted or packaged files are wrong
module_exists	Module already in system
min_redis_pack_version	Module isn't supported yet in this Redis pack
unsupported_module_capabilities	The module does not support required capabilities
os_not_supported	This module is not supported for this operating system
dependencies_not_supported	This endpoint does not support dependencies, see v2

Status codes

Code	Description
400 Bad Request	Either missing module file or an invalid module file.

Examples

cURL

```
$ curl -k -u "[username]:[password]" -X POST  
-F "module=@/tmp/rejson.Linux-ubuntu18.04-x86_64.2.0.8.zip"  
https://[host][:port]/v1/modules
```

Python

```
import requests  
  
url = "https://[host][:port]/v1/modules"  
  
files=[  
    ('module',  
     ('rejson.Linux-ubuntu18.04-x86_64.2.0.8.zip',  
      open('/tmp/rejson.Linux-ubuntu18.04-x86_64.2.0.8.zip','rb')),  
     'application/zip')  
]  
]  
auth=("[username]", "[password]")  
  
response = requests.request("POST", url,  
                            auth=auth, files=files, verify=False)  
  
print(response.text)
```

Upload module v2

POST /v2/modules

Asynchronously uploads a new module to the cluster.

The request must contain a Redis module bundled using [RedisModule Packer](#).

For modules in Redis Stack, download the module from the [Download Center](#). See [Install a module on a cluster](#) for more information.

Permissions

Permission name	Roles
update_cluster	admin

Request

Example HTTP request

POST /v2/modules

Headers

Key	Value	Description
Host	string	Domain name
Accept	*/*	Accepted media type
Content-Length	integer	Length of the request body in octets
Expect	100-continue	Requires particular server behaviors
Content-Type	multipart/form-data;	Media type of request/response body

Response

Returns a [module object](#) with an additional `action_uid` field.

You can use the `action_uid` to track the progress of the module upload.

Example JSON body

```
{  
    "action_uid": "dfc0152c-8449-4b1c-9184-480ea7cb526c",  
    "author": "RedisLabs",  
    "capabilities": [  
        "types",  
        "crdb",  
        "failover_migrate",  
        "persistence_aof",  
        "persistence_rdb",  
        "clustering",  
        "backup_restore"  
    ],  
    "command_line_args": "Plugin gears_python CreateVenv 1",  
    "config_command": "RG.CONFIGSET",  
    "dependencies": {  
        "gears_jvm": {  
            "sha256": "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9",  
            "url": "http://example.com/redisgears_plugins/jvm_plugin/gears-jvm.linux-centos7-x64.0.1.0.tgz"  
        },  
        "gears_python": {  
            "sha256": "22dca9cd75484cb15b8130db37f5284e22e3759002154361f72f6d2db46ee682",  
            "url": "http://example.com/redisgears-python.linux-centos7-x64.1.2.1.tgz"  
        }  
    },  
    "description": "Dynamic execution framework for your Redis data",  
    "display_name": "RedisGears",  
    "email": "user@example.com",  
    "homepage": "http://redisgears.io",  
    "is_bundled": false,  
    "license": "Redis Source Available License Agreement",  
    "min_redis_pack_version": "6.0.0",  
    "min_redis_version": "6.0.0",  
    "module_name": "rg",  
    "semantic_version": "1.2.1",  
    "sha256": "2935ea53611803c8acf0015253c5ae1cd81391bbacb23e14598841e1edd8d28b",  
    "uid": "98f255d5d33704c8e4e97897fd92e32d",  
    "version": 10201  
}
```

Error codes

The server may return a JSON object with `error_code` and `message` fields that provide additional information.

Possible `error_code` values include [/v1/modules error codes](#) and the following:

Code	Description
invalid_dependency_data	Provided dependencies have an unexpected format

Status codes

Code	Description
200 OK	Success, scheduled module upload.
400 Bad Request	Module name or version does not exist.
404 Not Found	Dependency not found.
500 Internal Server Error	Failed to get dependency.

Delete module v1

```
DELETE /v1/modules/{string: uid}
```

 | Note: DELETE /v1/modules is deprecated as of Redis Enterprise Software version 7.2. Use [DELETE /v2/modules](#) instead.

Delete a module.

Permissions

Permission name	Roles
update_cluster	admin

Request

Example HTTP request

```
DELETE /v1/modules/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The module's unique ID.

Response

Returns a status code to indicate module deletion success or failure.

Error codes

Code	Description
dependencies_not_supported	You can use the following API endpoint to delete this module with its dependencies: /v2/modules/<uid>

Status codes

Code	Description
200 OK	Success, the module is deleted.
404 Not Found	Attempting to delete a nonexistent module.
406 Not Acceptable	The request is not acceptable.

Delete module v2

```
DELETE /v2/modules/{string: uid}
```

Delete a module.

Permissions

Permission name	Roles
update_cluster	admin

Request

Example HTTP request

```
DELETE /v2/modules/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The module's unique ID.

Response

Returns a JSON object with an `action_uid` that allows you to track the progress of module deletion.

Status codes

Code	Description
200 OK	Success, scheduled module deletion.
404 Not Found	Attempting to delete a nonexistent module.

Updated: August 17, 2023

Configure module requests

Method	Path	Description
POST	/v1/modules/config/bdb/{uid}	Configure module

Configure module

```
POST /v1/modules/config/bdb/{string: uid}
```

Use the module runtime configuration command (if defined) to configure new arguments for the module.

Required permissions

Permission name
edit_bdb_module

Request

Example HTTP request

```
POST /modules/config/bdb/1
```

Example JSON body

```
{
  "modules": [
    {
      "module_name": "search",
      "module_args": "MINPREFIX 3 MAXEXPANSIONS 1000"
    }
  ]
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Field	Type	Description
modules	list of JSON objects	List of modules (module_name) and their new configuration settings (module_args)
module_name	search ReJSON graph timeseries bf	Module's name
module_args	string	Module command line arguments (pattern does not allow special characters &,<,>,")

Response

Returns a status code. If an error occurs, the response body may include an error code and message with more details.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
db_not_exist	Database with given UID doesn't exist in cluster
missing_field	"module_name" or "module_args" are not defined in request
invalid_schema	JSON object received is not a dict object
param_error	"module_args" parameter was not parsed properly
module_not_exist	Module with given "module_name" does not exist for the database

Status codes

Code	Description
200 OK	Success, module updated on bdb.

Code	Description
404 Not Found	bdb not found.
400 Bad Request	Bad or missing configuration parameters.
406 Not Acceptable	Module does not support runtime configuration of arguments.

Updated: February 23, 2023

Upgrade module requests

Method	Path	Description
POST	/v1/modules/upgrade/bdb/{uid}	Upgrade module

Upgrade module

POST /v1/modules/upgrade/bdb/{string: uid}

Upgrades the module version on a specific database.

Required permissions

Permission name

[edit_bdb_module](#)

Request

Example HTTP request

POST /modules/upgrade/bdb/1

Example JSON body

```
{
  "modules": [
    {"module_name": "ReJson",
     "current_semantic_version": "2.2.1",
     "new_module": "aa3648d79bd4082d414587c42ea0b234"}
  ],
  // Optional fields to fine-tune restart and failover behavior:
  "preserve_roles": true,
  "may_discard_data": false
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Field	Type	Description
-------	------	-------------

Field	Type	Description
modules	list	<p>List of dicts representing the modules that will be upgraded. Each dict must include:</p> <ul style="list-style-type: none"> • current_module: UID of a module to upgrade • new_module: UID of the module we want to upgrade to • new_module_args: args list for the new module
preserve_roles	boolean	Preserve shards' master/replica roles (optional)
may_discard_data	boolean	Discard data in a non-replicated non-persistent database (optional)

Response

Returns the upgraded [module object](#).

Example JSON body

```
{
  "uid": 1,
  "name": "name of database #1",
  "module_id": "aa3648d79bd4082d414587c42ea0b234",
  "module_name": "ReJson",
  "semantic_version": "2.2.2"
  // additional fields...
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
missing_module	Module is not present in cluster.
module_downgrade_unsupported	Module downgrade is not allowed.
redis_incompatible_version	Module <code>min_redis_version</code> is bigger than the current Redis version.
redis_pack_incompatible_version	Module <code>min_redis_pack_version</code> is bigger than the current Redis Enterprise version.
unsupported_module_capabilities	New version of module does support all the capabilities needed for the database configuration

Status codes

Code	Description
200 OK	Success, module updated on bdb.
404 Not Found	bdb or node not found.
400 Bad Request	Bad or missing configuration parameters.
406 Not Acceptable	The requested configuration is invalid.

Updated: November 29, 2021

Nodes requests

Method	Path	Description
GET	/v1/nodes	Get all cluster nodes
GET	/v1/nodes/{uid}	Get a single cluster node
PUT	/v1/nodes/{uid}	Update a node

Get all nodes

```
GET /v1/nodes
```

Get all cluster nodes.

Permissions

Permission name	Roles
view_all_nodes_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /nodes
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON array of [node objects](#).

Example JSON body

```
[
  {
    "uid": 1,
    "status": "active",
    "uptime": 262735,
    "total_memory": 6260334592,
    "software_version": "0.90.0-1",
    "ephemeral_storage_size": 20639797248,
    "persistent_storage_path": "/var/opt/redislabs/persist",
    "persistent_storage_size": 20639797248,
    "os_version": "Ubuntu 14.04.2 LTS",
    "ephemeral_storage_path": "/var/opt/redislabs/tmp",
    "architecture": "x86_64",
    "shard_count": 23,
    "public_addr": "",
    "cores": 4,
    "rack_id": "",
    "supported_database_versions": [
      {
        "db_type": "memcached",
        "version": "1.4.17"
      },
      {
        "db_type": "redis",
        "version": "2.6.16"
      },
      {
        "db_type": "redis",
        "version": "2.8.19"
      }
    ],
    "shard_list": [1, 3, 4],
    "addr": "10.0.3.61"
  },
  {
    "uid": 1,
    "status": "active",
    "// additional fields..."
  }
]
```

Status codes

Code	Description
200 OK	No error

Get node

```
GET /v1/nodes/{int: uid}
```

Get a single cluster node.

Permissions

Permission name	Roles
view_node_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /nodes/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the node requested.

Response

Returns a [node object](#).

Example JSON body

```
{
  "uid": 1,
  "name": "node:1",
  "// additional fields..."
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Node UID does not exist

Update node

```
PUT /v1/nodes/{int: uid}
```

Update a [node object](#).

Currently, you can edit the following attributes:

- addr
- external_addr
- recovery_path
- accept_servers



Note: You can only update the `addr` attribute for offline nodes. Otherwise, the request returns an error.

Permissions

Permission name	Roles
update_node	admin

Request

Example HTTP request

```
PUT /nodes/1
```

Example JSON body

```
{
  "addr": "10.0.0.1",
  "external_addr": [
    "192.0.2.24"
  ]
}
```

Request headers

Key	Value	Description
Host	cluster.fqdn	Domain name
Accept	application/json	Accepted media type
Content-Type	application/json	Media type of request/response body

URL parameters

Field	Type	Description
uid	integer	The unique ID of the updated node.

Body

Field	Type	Description
addr	string	Internal IP address of node
external_addr	JSON array	External IP addresses of the node
recovery_path	string	Path for recovery files
accept_servers	boolean	If true, no shards will be created on the node

Response

If the request is successful, the body will be empty. Otherwise, it may contain a JSON object with an error code and error message.

Status codes

Code	Description
200 OK	No error, the request has been processed.
406 Not Acceptable	Update request cannot be processed.
400 Bad Request	Bad content provided.

Error codes

Code	Description
node_not_found	Node does not exist
node_not_offline	Attempted to change node address while it is online
node_already_populated	The node contains shards or endpoints, cannot disable accept_servers

Code	Description
invalid_oss_cluster_port_mapping	Cannot enable “accept_servers” since there are databases with “oss_cluster_port_mapping” that do not have a port configuration for the current node
node_already_has_rack_id	Attempted to change node’s rack_id when it already has one

Updated: August 15, 2022

Node actions requests

Method	Path	Description
GET	/v1/nodes/actions	Get status of all actions on all nodes
GET	/v1/nodes/{node_uid}/actions	Get status of all actions on a specific node
GET	/v1/nodes/{node_uid}/actions/{action}	Get status of an action on a specific node
POST	/v1/nodes/{node_uid}/actions/{action}	Initiate node action
DELETE	/v1/nodes/{node_uid}/actions/{action}	Cancel action or remove action status

Get all actions

```
GET /v1/nodes/actions
```

Get the status of all currently executing, pending, or completed actions on all nodes.

Permissions

Permission name	Roles
view_status_of_all_node_actions	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /nodes/actions
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a list of [action objects](#).

Status codes

Code	Description
200 OK	No error, response provides details about an ongoing action.

Get node actions statuses

```
GET /v1/nodes/{node_uid}/actions
```

Get the status of all actions on a specific node.

Required permissions

Permission name	Roles
view_status_of_node_action	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /nodes/1/actions
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
action	string	The action to check.

Response

Returns a JSON object that includes a list of [action objects](#) for the specified node.

If no actions are available, the response will include an empty array.

Example JSON body

```
{
  "actions": [
    {
      "name": "remove_node",
      "node_uid": "1",
      "status": "running",
      "progress": 10
    }
  ]
}
```

Error codes

Code	Description
internal_error	An internal error that cannot be mapped to a more precise error code has been encountered.

Code	Description
insufficient_resources	The cluster does not have sufficient resources to complete the required operation.

Status codes

Code	Description
200 OK	No error, response provides details about an ongoing action.
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run).

Get node action status

```
GET /v1/nodes/{node_uid}/actions/{action}
```

Get the status of a currently executing, queued, or completed action on a specific node.

Request

Example HTTP request

```
GET /nodes/1/actions/remove
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns an [action object](#) for the specified node.

Error codes

Code	Description
internal_error	An internal error that cannot be mapped to a more precise error code has been encountered.
insufficient_resources	The cluster does not have sufficient resources to complete the required operation.

Status codes

Code	Description
200 OK	No error, response provides details about an ongoing action.
404 Not Found	Action does not exist (i.e. not currently running and no available status of last run).

Initiate node action

```
POST /v1/nodes/{node_uid}/actions/{action}
```

Initiate a node action.

The API allows only a single instance of any action type to be invoked at the same time, and violations of this requirement will result in a 409 CONFLICT

response.

The caller is expected to query and process the results of the previously executed instance of the same action, which will be removed as soon as the new one is submitted.

Permissions

Permission name	Roles
start_node_action	admin

Request

Example HTTP request

```
POST /nodes/1/actions/remove
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
action	string	The name of the action required.

Currently supported actions are:

- remove: Removes the node from the cluster after migrating all bound resources to other nodes. As soon as a successful remove request is received, the cluster will no longer automatically migrate resources (shards/endpoints) to the node, even if the remove task fails at some point.
- maintenance_on: Creates a snapshot of the node, migrates shards to other nodes, and prepares the node for maintenance. See[maintenance mode](#) for more information.
 - If there aren't enough resources to migrate shards out of the maintained node, set "keep_slave_shards" : true in the request body to keep the replica shards in place but demote any master shards.
- maintenance_off: Restores node to its previous state before maintenance started. See[maintenance mode](#) for more information.
 - By default, it uses the latest node snapshot.
 - Use "snapshot_name" : " . ." in the request body to restore the state from a specific snapshot.
 - To avoid restoring shards at the node, use "skip_shards_restore" : true.
- enslave_node: Turn node into a replica.

Response

The body content may provide additional action details.

Status codes

Code	Description
200 OK	Action initiated successfully.
409 Conflict	Only a single instance of any action type can be invoked at the same time.

Example requests

cURL

```
$ curl -k -X POST -u "[username]:[password]" -d "{}"  
https://[host][:port]/v1/nodes/1/actions/remove
```

Python

```
import requests
import json

url = "https://[host][port]/v1/nodes/1/actions/remove"

payload = json.dumps({})
headers = {
    'Content-Type': 'application/json',
}
auth = ("[username]", "[password]")

response = requests.request("POST", url, auth=auth, headers=headers, data=payload)

print(response.text)
```

Cancel action

```
DELETE /v1/nodes/{node_uid}/actions/{action}
```

Cancel a queued or executing node action, or remove the status of a previously executed and completed action.

Permissions

Permission name

[cancel_node_action](#)

Request

Example HTTP request

```
DELETE /nodes/1/actions/remove
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
action	string	The name of the action to cancel.

Response

Returns a status code.

Status codes

Code	Description
200 OK	Action will be cancelled when possible.
404 Not Found	Action unknown or not currently running.

Node alerts requests

Method	Path	Description
GET	/v1/nodes/alerts	Get all alert states for all nodes
GET	/v1/nodes/alerts/{uid}	Get all alert states for a node
GET	/v1/nodes/alerts/{uid}/{alert}	Get node alert state

Get all alert states

GET /v1/nodes/alerts

Get all alert states for all nodes.

Required permissions

Permission name

[view_all_nodes_alerts](#)

Request

Example HTTP request

GET /nodes/alerts

Request headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
ignore_settings	boolean	Retrieve updated alert state regardless of the cluster's alert_settings. When not present, a disabled alert will always be retrieved as disabled with a false state. (optional)

Response

Returns a hash of node UIDs and the [alert states](#) for each node.

Example JSON body

```
{
  "1": {
    "node_cpu_utilization": {
      "change_time": "2014-12-22T10:42:00Z",
      "change_value": {
        "cpu_util": 2.50000000145519,
        "global_threshold": "1",
        "state": true
      },
      "enabled": true,
      "state": true,
      "severity": "WARNING"
    },
    "..."
  },
  ...
}
```

Status codes

Code	Description
200 OK	No error

Get node alert states

GET /v1/nodes/alerts/{int: uid}

Get all alert states for a node.

Required permissions

Permission name
view_node_alerts

Request

Example HTTP request

GET /nodes/alerts/1

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
ignore_settings	boolean	Retrieve updated alert state regardless of the cluster's alert_settings. When not present, a disabled alert will always be retrieved as disabled with a false state. (optional)

Response

Returns a hash of [alert objects](#) and their states for a specific node.

Example JSON body

```
{  
  "node_cpu_utilization": {  
    "change_time": "2014-12-22T10:42:00Z",  
    "change_value": {  
      "cpu_util": 2.50000000145519,  
      "global_threshold": "1",  
      "state": true  
    },  
    "enabled": true,  
    "state": true,  
    "severity": "WARNING",  
  },  
  "..."  
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified node does not exist

Get node alert state

GET /v1/nodes/alerts/{int: uid}/{alert}

Get a node alert state.

Required permissions

Permission name
view_node_alerts

Request

Example HTTP request

GET /nodes/alerts/1/node_cpu_utilization

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
ignore_settings	boolean	Retrieve updated alert state regardless of the cluster's alert_settings. When not present, a disabled alert will always be retrieved as disabled with a false state. (optional)

Response

Returns an [alert object](#).

Example JSON body

```
{  
  "change_time": "2014-12-22T10:42:00Z",  
  "change_value": {  
    "cpu_util": 2.50000000145519,  
    "global_threshold": "1",  
    "state": true  
  },  
  "enabled": true,  
  "state": true,  
  "severity": "WARNING",  
}  
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Specified alert or node does not exist

Updated: November 29, 2021

Node stats requests

Method	Path	Description
GET	/v1/nodes/stats	Get stats for all nodes
GET	/v1/nodes/stats/{uid}	Get stats for a single node

Get all nodes stats

```
GET /v1/nodes/stats
```

Get statistics for all nodes.

Required permissions

Permission name

[view_all_nodes_stats](#)

Request

Example HTTP request

```
GET /nodes/stats?interval=1hour&stime=2014-08-28T10:00:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name

Key	Value	Description
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns a JSON array of [statistics](#) for all nodes.

Example JSON body

```
[
  {
    "uid": "1",
    "intervals": [
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:40:11Z",
        "etime": "2015-05-28T08:40:12Z",
        "conns": 0.0,
        "cpu_idle": 0.549999999983585,
        "cpu_system": 0.0349999999985448,
        "cpu_user": 0.3800000000101863,
        "egress_bytes": 0.0,
        "ephemeral_storage_avail": 2929315840.0,
        "ephemeral_storage_free": 3977830400.0,
        "free_memory": 893485056.0,
        "ingress_bytes": 0.0,
        "persistent_storage_avail": 2929315840.0,
        "persistent_storage_free": 3977830400.0,
        "total_req": 0.0
      },
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:40:12Z",
        "etime": "2015-05-28T08:40:13Z",
        "cpu_idle": 1.2,
        "// additional fields..."
      }
    ]
  },
  {
    "uid": "2",
    "intervals": [
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:40:11Z",
        "etime": "2015-05-28T08:40:12Z",
        "conns": 0.0,
        "cpu_idle": 0.549999999983585,
        "cpu_system": 0.0349999999985448,
        "cpu_user": 0.3800000000101863,
        "egress_bytes": 0.0,
        "ephemeral_storage_avail": 2929315840.0,
        "ephemeral_storage_free": 3977830400.0,
        "free_memory": 893485056.0,
        "ingress_bytes": 0.0,
        "persistent_storage_avail": 2929315840.0,
        "persistent_storage_free": 3977830400.0,
        "total_req": 0.0
      },
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:40:12Z",
        "etime": "2015-05-28T08:40:13Z",
        "cpu_idle": 1.2,
        "// additional fields..."
      }
    ]
  }
]
```

Status codes

Code	Description
200 OK	No error

Code	Description
404 Not Found	No nodes exist

Get node stats

```
GET /v1/nodes/stats/{int: uid}
```

Get statistics for a node.

Required permissions

Permission name

[view_node_stats](#)

Request

Example HTTP request

```
GET /nodes/stats/1?interval=1hour&stime=2014-08-28T10:00:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the node requested.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for the specified node.

Example JSON body

```
{
  "uid": "1",
  "intervals": [
    {
      "interval": "1sec",
      "stime": "2015-05-28T08:40:11Z",
      "etime": "2015-05-28T08:40:12Z",
      "conns": 0.0,
      "cpu_idle": 0.549999999983585,
      "cpu_system": 0.0349999999985448,
      "cpu_user": 0.3800000000101863,
      "egress_bytes": 0.0,
      "ephemeral_storage_avail": 2929315840.0,
      "ephemeral_storage_free": 3977830400.0,
      "free_memory": 893485056.0,
      "ingress_bytes": 0.0,
      "persistent_storage_avail": 2929315840.0,
      "persistent_storage_free": 3977830400.0,
      "total_req": 0.0
    },
    {
      "interval": "1sec",
      "stime": "2015-05-28T08:40:12Z",
      "etime": "2015-05-28T08:40:13Z",
      "cpu_idle": 1.2,
      "// additional fields..."
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Node does not exist
406 Not Acceptable	Node isn't currently active
503 Service Unavailable	Node is in recovery state

Updated: August 15, 2022

Latest node stats requests

Method	Path	Description
GET	/v1/nodes/stats/last	Get latest stats for all nodes
GET	/v1/nodes/stats/last/{uid}	Get latest stats for a single node

Get latest stats for all nodes

```
GET /v1/nodes/stats/last
```

Get latest statistics for all nodes.

Required permissions

Permission name
view_all_nodes_stats

Request

Example HTTP request

```
GET /nodes/stats/last?interval=1sec&stime=2015-10-14T06:29:43Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week. Default: 1sec. (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns most recent [statistics](#) for all nodes.

Example JSON body

```
{
  "1": {
    "conns": 0.0,
    "cpu_idle": 0.922500000015134,
    "cpu_system": 0.00749999999708962,
    "cpu_user": 0.0174999999810825,
    "cur_aof_rewrites": 0.0,
    "egress_bytes": 7887.0,
    "ephemeral_storage_avail": 75821363200.0,
    "ephemeral_storage_free": 81189969920.0,
    "etime": "2015-10-14T06:29:44Z",
    "free_memory": 2956963840.0,
    "ingress_bytes": 4950.0,
    "interval": "1sec",
    "persistent_storage_avail": 75821363200.0,
    "persistent_storage_free": 81189969920.0,
    "stime": "2015-10-14T06:29:43Z",
    "total_req": 0.0
  },
  "2": {
    "conns": 0.0,
    "cpu_idle": 0.922500000015134,
    "// additional fields...
  }
}
```

Status codes

Code	Description
------	-------------

Code	Description
200 OK	No error
404 Not Found	No nodes exist

Get latest node stats

```
GET /v1/nodes/stats/last/{int: uid}
```

Get the latest statistics of a node.

Required permissions

Permission name

[view_node_stats](#)

Request

Example HTTP request

```
GET /nodes/stats/last/1?interval=1sec&stime=2015-10-13T09:01:54Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the node requested.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week. Default: 1sec. (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns the most recent [statistics](#) for the specified node.

Example JSON body

```
{
  "1": {
    "conns": 0.0,
    "cpu_idle": 0.8049999999930151,
    "cpu_system": 0.02750000000014552,
    "cpu_user": 0.1200000000080036,
    "cur_aof_rewrites": 0.0,
    "egress_bytes": 2169.0,
    "ephemeral_storage_avail": 75920293888.0,
    "ephemeral_storage_free": 81288900608.0,
    "etime": "2015-10-13T09:01:55Z",
    "free_memory": 3285381120.0,
    "ingress_bytes": 3020.0,
    "interval": "1sec",
    "persistent_storage_avail": 75920293888.0,
    "persistent_storage_free": 81288900608.0,
    "stime": "2015-10-13T09:01:54Z",
    "total_req": 0.0
  }
}
```

Error codes

Code	Description
200 OK	No error
404 Not Found	Node does not exist
406 Not Acceptable	Node isn't currently active
503 Service Unavailable	Mode is in recovery state

Updated: August 15, 2022

Node status requests

Method	Path	Description
GET	/v1/nodes/status	Get the status of all nodes
GET	/v1/nodes/{uid}/status	Get a node's status

Get all node statuses

GET /v1/nodes/status

Gets the status of all nodes. Includes each node's hostname and role in the cluster:

- Primary nodes return "role": "master"
- Replica nodes return "role": "slave"

Required permissions

Permission name

view_node_info

Request

Example HTTP request

GET /nodes/status

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

For each node in the cluster, returns a JSON object that contains the node's hostname and role.

Example JSON body

```
{  
  "1": {  
    "hostname": "3d99db1fdf4b",  
    "role": "master"  
  },  
  "2": {  
    "hostname": "fc7a3d332458",  
    "role": "slave"  
  },  
  "3": {  
    "hostname": "b87cc06c830f",  
    "role": "slave"  
  }  
}
```

Status codes

Code	Description
200 OK	No error

Get node status

GET /v1/nodes/{int: uid}/status

Gets the status of a node. Includes the node's hostname and role in the cluster:

- Primary nodes return "role": "master"
- Replica nodes return "role": "slave"

Required permissions

Permission name

[view_node_info](#)

Request

Example HTTP request

GET /nodes/1/status

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The node's unique ID.

Response

Returns a JSON object that contains the node's hostname and role.

Example JSON body

```
{
  "hostname": "3d99db1fdf4b",
  "role": "master"
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Node UID does not exist

Updated: January 11, 2023

OCSP requests

Method	Path	Description
GET	/v1/ocsp	Get OCSP configuration
PUT	/v1/ocsp	Update OCSP configuration

Get OCSP configuration

GET /v1/ocsp

Gets the cluster's OCSP configuration.

Required permissions

Permission name
view_ocsp_config

Request

Example HTTP request

GET /ocsp

Request headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns an [OCSP configuration object](#).

Example JSON body

```
{  
    "ocsp_functionality": true,  
    "responder_url": "http://responder.ocsp.url.com",  
    "query_frequency": 3800,  
    "response_timeout": 2,  
    "recovery_frequency": 80,  
    "recovery_max_tries": 20  
}
```

Error codes

When errors occur, the server returns a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
ocsp_unsupported_by_capability	Not all nodes support OCSP capability

Status codes

Code	Description
200 OK	Success
406 Not Acceptable	Feature not supported in all nodes

Update OCSP configuration

PUT /v1/ocsp

Updates the cluster's OCSP configuration.

Required permissions

Permission name

[config_ocsp](#)

Request

Example HTTP request

PUT /ocsp

Example JSON body

```
{
  "ocsp_functionality": true,
  "query_frequency": 3800,
  "response_timeout": 2,
  "recovery_frequency": 80,
  "recovery_max_tries": 20
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include an [OCSP configuration object](#) with updated fields in the request body.

Response

Returns the updated [OCSP configuration object](#).

Error codes

When errors occur, the server returns a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
invalid_schema	An illegal parameter or a parameter with an illegal value
no_responder_url	Tried to enable OCSP with no responder URL configured
ocsp_unsupported_by_capability	Not all nodes support OCSP capability

Status codes

Code	Description
200 OK	Success, OCSP config has been set
400 Bad Request	Bad or missing configuration parameters
406 Not Acceptable	Feature not supported in all nodes

Updated: August 10, 2022

OCSP status requests

Method	Path	Description
GET	/v1/ocsp/status	Get OCSP status

Get OCSP status

GET /v1/ocsp/status

Gets the latest cached status of the proxy certificate's OCSP response.

Required permissions

Permission name

[view_ocsp_status](#)

Request

Example HTTP request

```
GET /ocsp/status
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns an [OCSP status object](#).

Example JSON body

```
{
  "responder_url": "http://responder.ocsp.url.com",
  "cert_status": "REVOKED",
  "produced_at": "Wed, 22 Dec 2021 12:50:11 GMT",
  "this_update": "Wed, 22 Dec 2021 12:50:11 GMT",
  "next_update": "Wed, 22 Dec 2021 14:50:00 GMT",
  "revocation_time": "Wed, 22 Dec 2021 12:50:04 GMT"
}
```

Error codes

When errors occur, the server returns a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
<code>ocsp_unsupported_by_capability</code>	Not all nodes support OCSP capability
<code>invalid_ocsp_response</code>	The server returned a response that is not OCSP-compatible

Status codes

Code	Description
<code>200 OK</code>	Success
<code>406 Not Acceptable</code>	Feature not supported in all nodes

Updated: August 10, 2022

OCSP test requests

Method	Path	Description
<code>POST</code>	<code>/v1/ocsp/test</code>	Test OCSP

Test OCSP

POST /v1/ocsp/test

Queries the OCSP server for the proxy certificate's latest status and returns the response as JSON. It caches the response if the OCSP feature is enabled.

Required permissions

Permission name

test_ocsp_status

Request

Example HTTP request

POST /ocsp/test

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns an [OCSP status object](#).

Example JSON body

```
{  
    "responder_url": "http://responder.ocsp.url.com",  
    "cert_status": "REVOKED",  
    "produced_at": "Wed, 22 Dec 2021 12:50:11 GMT",  
    "this_update": "Wed, 22 Dec 2021 12:50:11 GMT",  
    "next_update": "Wed, 22 Dec 2021 14:50:00 GMT",  
    "revocation_time": "Wed, 22 Dec 2021 12:50:04 GMT"  
}
```

Error codes

When errors occur, the server returns a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
no_responder_url	Tried to test OCSP status with no responder URL configured
ocsp_unsupported_by_capability	Not all nodes support OCSP capability
task_queued_for_too_long	OCSP polling task was in status “queued” for over 5 seconds
invalid_ocsp_response	The server returned a response that is not compatible with OCSP

Status codes

Code	Description
200 OK	Success querying the OCSP server
406 Not Acceptable	Feature is not supported in all nodes
500 Internal Server Error	responder_url is not configured or polling task failed

Redis access control list (ACL) requests

Method	Path	Description
GET	/v1/redis_acls	Get all Redis ACLs
GET	/v1/redis_acls/{uid}	Get a single Redis ACL
PUT	/v1/redis_acls/{uid}	Update a Redis ACL
POST	/v1/redis_acls	Create a new Redis ACL
DELETE	/v1/redis_acls/{uid}	Delete a Redis ACL

Get all Redis ACLs

```
GET /v1/redis_acls
```

Get all Redis ACL objects.

Permissions

Permission name	Roles
view_all_redis_acls_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /redis_acls
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON array of [Redis ACL objects](#).

Example JSON body

```
[
  {
    "uid": 1,
    "name": "Full Access",
    "acl": "+@all ~*"
  },
  {
    "uid": 2,
    "name": "Read Only",
    "acl": "+@read ~*"
  },
  {
    "uid": 3,
    "name": "Not Dangerous",
    "acl": "+@all -@dangerous ~*"
  },
  {
    "uid": 17,
    "name": "Geo",
    "acl": "~* +@geo"
  }
]
```

Status codes

Code	Description
200 OK	No error
501 Not Implemented	Cluster doesn't support redis_acl yet.

Get Redis ACL

```
GET /v1/redis_acls/{int: uid}
```

Get a single Redis ACL object.

Permissions

Permission name	Roles
view_redis_acl_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /redis_acls/1
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The object's unique ID.

Response

Returns a [Redis ACL object](#).

Example JSON body

```
{
  "uid": 17,
  "name": "Geo",
  "acl": "~* +@geo"
}
```

Status codes

Code	Description
200 OK	Success.
403 Forbidden	Operation is forbidden.
404 Not Found	redis_acl does not exist.
501 Not Implemented	Cluster doesn't support redis_acl yet.

Update Redis ACL

```
PUT /v1/redis_acls/{int: uid}
```

Update an existing Redis ACL object.

Permissions

Permission name	Roles
update_redis_acl	admin

Request

Example HTTP request

```
PUT /redis_acls/17
```

Example JSON body

```
{
  "acl": "~* +@geo -@dangerous"
}
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [Redis ACL object](#) with updated fields in the request body.

Response

Returns the updated [Redis ACL object](#).

Example JSON body

```
{  
  "uid": 17,  
  "name": "Geo",  
  "acl": "~* +@geo -@dangerous"  
}
```

Error codes

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists
invalid_param	A parameter has an illegal value

Status codes

Code	Description
200 OK	Success, redis_acl was updated.
400 Bad Request	Bad or missing configuration parameters.
404 Not Found	Attempting to change a non-existent redis_acl.
409 Conflict	Cannot change a read-only object
501 Not Implemented	Cluster doesn't support redis_acl yet.

Create Redis ACL

```
POST /v1/redis_acls
```

Create a new Redis ACL object.

Permissions

Permission name	Roles
create_redis_acl	admin

Request

Example HTTP request

```
POST /redis_acls
```

Example JSON body

```
{  
  "name": "Geo",  
  "acl": "~* +@geo"  
}
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [Redis ACL object](#) in the request body.

Response

Returns the newly created [Redis ACL object](#).

Example JSON body

```
{  
    "uid": 17,  
    "name": "Geo",  
    "acl": "~* +@geo"  
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists
missing_field	A needed field is missing
invalid_param	A parameter has an illegal value

Status codes

Code	Description
200 OK	Success, redis_acl is created.
400 Bad Request	Bad or missing configuration parameters.
501 Not Implemented	Cluster doesn't support redis_acl yet.

Examples

cURL

```
curl -k -u "[username]:[password]" -X POST \  
-H 'Content-Type: application/json' \  
-d '{ "name": "Geo", "acl": "~* +@geo" }' \  
https://[host][:port]/v1/redis_acls
```

Python

```

import requests
import json

url = "https://[host][:port]/v1/redis_acls"

headers = {
    'Content-Type': 'application/json'
}

payload = json.dumps({
    "name": "Geo",
    "acl": "~* +@geo"
})
auth=("[username]", "[password]")

response = requests.request("POST", url,
                            auth=auth, headers=headers, payload=payload, verify=False)

print(response.text)

```

Delete Redis ACL

```
DELETE /v1/redis_acls/{int: uid}
```

Delete a Redis ACL object.

Permissions

Permission name	Roles
delete_redis_acl	admin

Request

Example HTTP request

```
DELETE /redis_acls/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The redis_acl unique ID.

Response

Returns a status code that indicates the Redis ACL deletion success or failure.

Status codes

Code	Description
200 OK	Success, the redis_acl is deleted.

Code	Description
406 Not Acceptable	The request is not acceptable.
409 Conflict	Cannot delete a read-only object
501 Not Implemented	Cluster doesn't support redis_acl yet.

Updated: August 18, 2022

Roles requests

Method	Path	Description
GET	/v1/roles	Get all roles
GET	/v1/roles/{uid}	Get a single role
PUT	/v1/roles/{uid}	Update an existing role
POST	/v1/roles	Create a new role
DELETE	/v1/roles/{uid}	Delete a role

Get all roles

```
GET /v1/roles
```

Get all roles' details.

Permissions

Permission name	Roles
view_all_roles_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /roles
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON array of [role objects](#).

Example JSON body

```
[
  {
    "uid": 1,
    "name": "Admin",
    "management": "admin"
  },
  {
    "uid": 2,
    "name": "Cluster Member",
    "management": "cluster_member"
  },
  {
    "uid": 3,
    "name": "Cluster Viewer",
    "management": "cluster_viewer"
  },
  {
    "uid": 4,
    "name": "DB Member",
    "management": "db_member"
  },
  {
    "uid": 5,
    "name": "DB Viewer",
    "management": "db_viewer"
  },
  {
    "uid": 6,
    "name": "None",
    "management": "none"
  },
  {
    "uid": 17,
    "name": "DBA",
    "management": "admin"
  }
]
```

Status codes

Code	Description
200 OK	No error
501 Not Implemented	Cluster doesn't support roles yet.

Get role

```
GET /v1/roles/{int: uid}
```

Get the details of a single role.

Permissions

Permission name	Roles
view_role_info	admin cluster_member cluster_viewer db_member db_viewer

Request

Example HTTP request

```
GET /roles/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The role's unique ID.

Response

Returns a [role object](#).

Example JSON body

```
{
  "uid": 17,
  "name": "DBA",
  "management": "admin"
}
```

Status codes

Code	Description
200 OK	Success.
403 Forbidden	Operation is forbidden.
404 Not Found	Role does not exist.
501 Not Implemented	Cluster doesn't support roles yet.

Update role

```
PUT /v1/roles/{int: uid}
```

Update an existing role's details.

Permissions

Permission name	Roles
update_role	admin

Request

Example HTTP request

```
PUT /roles/17
```

Example JSON body

```
{
  "management": "cluster_member"
}
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Body

Include a [role object](#) with updated fields in the request body.

Response

Returns a [role object](#) with the updated fields.

Example JSON body

```
{
  "uid": 17,
  "name": "DBA",
  "management": "cluster_member"
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists.
change_last_admin_role_not_allowed	At least one user with admin role should exist.

Status codes

Code	Description
200 OK	Success, role is created.
400 Bad Request	Bad or missing configuration parameters.
404 Not Found	Attempting to change a non-existent role.
501 Not Implemented	Cluster doesn't support roles yet.

Create role

```
POST /v1/roles
```

Create a new role.

Permissions

Permission name	Roles
create_role	admin

Request

Example HTTP request

```
POST /roles
```

Example JSON body

```
{  
  "name": "DBA",  
  "management": "admin"  
}
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Body

Include a [role object](#) in the request body.

Response

Returns the newly created [role object](#).

Example JSON body

```
{  
  "uid": 17,  
  "name": "DBA",  
  "management": "admin"  
}
```

Error codes

Possible error_code values:

Code	Description
unsupported_resource	The cluster is not yet able to handle this resource type. This could happen in a partially upgraded cluster, where some of the nodes are still on a previous version.
name_already_exists	An object of the same type and name exists
missing_field	A needed field is missing

Status codes

Code	Description
200 OK	Success, role is created.
400 Bad Request	Bad or missing configuration parameters.
501 Not Implemented	Cluster doesn't support roles yet.

Examples

cURL

```
curl -k -u "[username]:[password]" -X POST \
-H 'Content-Type: application/json' \
-d '{ "name": "DBA", "management": "admin" }' \
https://[host][:port]/v1/roles
```

Python

```
import requests
import json

url = "https://[host][:port]/v1/roles"

headers = {
    'Content-Type': 'application/json'
}

payload = json.dumps({
    "name": "DBA",
    "management": "admin"
})
auth=("[username]", "[password]")

response = requests.request("POST", url,
    auth=auth, headers=headers, payload=payload, verify=False)

print(response.text)
```

Delete role

```
DELETE /v1/roles/{int: uid}
```

Delete a role object.

Permissions

Permission name	Roles
delete_role	admin

Request

Example HTTP request

```
DELETE /roles/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The role unique ID.

Response

Returns a status code to indicate role deletion success or failure.

Status codes

Code	Description
200 OK	Success, the role is deleted.
404 Not Found	Role does not exist.
406 Not Acceptable	The request is not acceptable.
501 Not Implemented	Cluster doesn't support roles yet.

Updated: August 10, 2022

Shards stats requests

Method	Path	Description
GET	/v1/shards/stats	Get stats for all shards
GET	/v1/shards/stats/{uid}	Get stats for a specific shard

Get all shards stats

GET /v1/shards/stats

Get statistics for all shards.

Required permissions

Permission name

view_all_shard_stats

Request

Example HTTP request

```
GET /shards/stats?interval=1hour&stime=2014-08-28T10:00:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
parent_uid	integer	Only return shard from the given BDB ID (optional)
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)

Field	Type	Description
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)
metrics	list	Comma-separated list of metric names for which we want statistics (default is all) (optional)

Response

Returns a JSON array of [statistics](#) for all shards.

Example JSON body

```
[
  {
    "status": "active",
    "uid": "1",
    "node_uid": "1",
    "assigned_slots": "0-8191",
    "intervals": [
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:27:35Z",
        "etime": "2015-05-28T08:27:40Z",
        "used_memory_peak": 5888264.0,
        "used_memory_rss": 5888264.0,
        "read_hits": 0.0,
        "pubsub_patterns": 0.0,
        "no_of_keys": 0.0,
        "mem_size_lua": 35840.0,
        "last_save_time": 1432541051.0,
        "sync_partial_ok": 0.0,
        "connected_clients": 9.0,
        "avg_ttl": 0.0,
        "write_misses": 0.0,
        "used_memory": 5651440.0,
        "sync_full": 0.0,
        "expired_objects": 0.0,
        "total_req": 0.0,
        "blocked_clients": 0.0,
        "pubsub_channels": 0.0,
        "evicted_objects": 0.0,
        "no_of_expires": 0.0,
        "interval": "1sec",
        "write_hits": 0.0,
        "read_misses": 0.0,
        "sync_partial_err": 0.0,
        "rdb_changes_since_last_save": 0.0
      },
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:27:40Z",
        "etime": "2015-05-28T08:27:45Z",
        "// additional fields..."
      }
    ]
  },
  {
    "uid": "2",
    "status": "active",
    "node_uid": "1",
    "assigned_slots": "8192-16383",
    "intervals": [
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:27:35Z",
        "etime": "2015-05-28T08:27:40Z",
        "// additional fields..."
      },
      {
        "interval": "1sec",
        "stime": "2015-05-28T08:27:40Z",
        "etime": "2015-05-28T08:27:45Z",
        "// additional fields..."
      }
    ]
  }
]
```

Status codes

Code	Description
200 OK	No error
404 Not Found	No shards exist

Get shard stats

GET /v1/shards/stats/{int: uid}

Get statistics for a specific shard.

Required permissions

Permission name

[view_shard_stats](#)

Request

Example HTTP request

```
GET /shards/stats/1?interval=1hour&stime=2014-08-28T10:00:00Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the shard requested.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns [statistics](#) for the specified shard.

Example JSON body

```
{
  "uid": "1",
  "status": "active",
  "node_uid": "1",
  "role": "master",
  "intervals": [
    {
      "interval": "1sec",
      "stime": "2015-05-28T08:24:13Z",
      "etime": "2015-05-28T08:24:18Z",
      "avg_ttl": 0.0,
      "blocked_clients": 0.0,
      "connected_clients": 9.0,
      "etime": "2015-05-28T08:24:18Z",
      "evicted_objects": 0.0,
      "expired_objects": 0.0,
      "last_save_time": 1432541051.0,
      "used_memory": 5651440.0,
      "mem_size_lua": 35840.0,
      "used_memory_peak": 5888264.0,
      "used_memory_rss": 5888264.0,
      "no_of_expires": 0.0,
      "no_of_keys": 0.0,
      "pubsub_channels": 0.0,
      "pubsub_patterns": 0.0,
      "rdb_changes_since_last_save": 0.0,
      "read_hits": 0.0,
      "read_misses": 0.0,
      "stime": "2015-05-28T08:24:13Z",
      "sync_full": 0.0,
      "sync_partial_err": 0.0,
      "sync_partial_ok": 0.0,
      "total_req": 0.0,
      "write_hits": 0.0,
      "write_misses": 0.0
    },
    {
      "interval": "1sec",
      "stime": "2015-05-28T08:24:18Z",
      "etime": "2015-05-28T08:24:23Z",
      "// additional fields..."
    }
  ]
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Shard does not exist
406 Not Acceptable	Shard isn't currently active

Updated: November 29, 2021

Latest shards stats requests

Method	Path	Description
GET	/v1/shards/stats/last	Get most recent stats for all shards

Method	Path	Description
GET	/v1/shards/stats/last/{uid}	Get most recent stats for a specific shard

Get latest stats for all shards

GET /v1/shards/stats/last

Get most recent statistics for all shards.

Required permissions

Permission name

[view_all_shard_stats](#)

Request

Example HTTP request

```
GET /shards/stats/last?interval=1sec&stime=015-05-27T08:27:35Z
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week. Default: 1sec (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns most recent [statistics](#) for all shards.

Example JSON body

```
{
  "1": {
    "interval": "1sec",
    "stime": "2015-05-28T08:27:35Z",
    "etime": "2015-05-28T08:28:36Z",
    "used_memory_peak": 5888264.0,
    "used_memory_rss": 5888264.0,
    "read_hits": 0.0,
    "pubsub_patterns": 0.0,
    "no_of_keys": 0.0,
    "mem_size_lua": 35840.0,
    "last_save_time": 1432541051.0,
    "sync_partial_ok": 0.0,
    "connected_clients": 9.0,
    "avg_ttl": 0.0,
    "write_misses": 0.0,
    "used_memory": 5651440.0,
    "sync_full": 0.0,
    "expired_objects": 0.0,
    "total_req": 0.0,
    "blocked_clients": 0.0,
    "pubsub_channels": 0.0,
    "evicted_objects": 0.0,
    "no_of_expires": 0.0,
    "interval": "1sec",
    "write_hits": 0.0,
    "read_misses": 0.0,
    "sync_partial_err": 0.0,
    "rdb_changes_since_last_save": 0.0
  },
  "2": {
    "interval": "1sec",
    "stime": "2015-05-28T08:27:40Z",
    "etime": "2015-05-28T08:28:45Z",
    "// additional fields..."
  }
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	No shards exist

Get latest shard stats

GET /v1/shards/stats/last/{int: uid}

Get most recent statistics for a specific shard.

Required permissions

Permission name
view_shard_stats

Request

Example HTTP request

GET /shards/stats/last/1?interval=1sec&stime=2015-05-28T08:27:35Z

Request headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The unique ID of the shard requested.

Query parameters

Field	Type	Description
interval	string	Time interval for which we want stats: 1sec/10sec/5min/15min/1hour/12hour/1week. Default: 1sec. (optional)
stime	ISO_8601	Start time from which we want the stats. Should comply with the ISO_8601 format (optional)
etime	ISO_8601	End time after which we don't want the stats. Should comply with the ISO_8601 format (optional)

Response

Returns the most recent [statistics](#) for the specified shard.

Example JSON body

```
{
  "1": {
    "interval": "1sec",
    "stime": "2015-05-28T08:27:35Z",
    "etime": "2015-05-28T08:27:36Z",
    "used_memory_peak": 5888264.0,
    "used_memory_rss": 5888264.0,
    "read_hits": 0.0,
    "pubsub_patterns": 0.0,
    "no_of_keys": 0.0,
    "mem_size_lua": 35840.0,
    "last_save_time": 1432541051.0,
    "sync_partial_ok": 0.0,
    "connected_clients": 9.0,
    "avg_ttl": 0.0,
    "write_misses": 0.0,
    "used_memory": 5651440.0,
    "sync_full": 0.0,
    "expired_objects": 0.0,
    "total_req": 0.0,
    "blocked_clients": 0.0,
    "pubsub_channels": 0.0,
    "evicted_objects": 0.0,
    "no_of_expires": 0.0,
    "interval": "1sec",
    "write_hits": 0.0,
    "read_misses": 0.0,
    "sync_partial_err": 0.0,
    "rdb_changes_since_last_save": 0.0
  }
}
```

Status codes

Code	Description
200 OK	No error
404 Not Found	Shard does not exist
406 Not Acceptable	Shard isn't currently active

Updated: November 29, 2021

Suffix requests

Method	Path	Description
GET	/v1/suffix/{name}	Get a single DNS suffix

Get suffix

GET /v1/suffix/{string: name}

Get a single DNS suffix.

Request

Example HTTP request

GET /suffix/cluster.fqdn

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
name	string	The unique name of the suffix requested.

Response

Returns a [suffix object](#).

Example JSON body

```
{  
  "name": "cluster.fqdn",  
  "// additional fields..."  
}
```

Status codes

Code	Description
------	-------------

Code	Description
200 OK	No error
404 Not Found	Suffix name does not exist

Updated: November 29, 2021

Suffixes requests

Method	Path	Description
GET	/v1/suffixes	Get all DNS suffixes

Get all suffixes

GET /v1/suffixes

Get all DNS suffixes in the cluster.

Request

Example HTTP request

GET /suffixes

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

The response body contains a JSON array with all suffixes, represented as [suffix objects](#).

Example JSON body

```
[  
  {  
    "name": "cluster.fqdn",  
    "// additional fields..."  
  },  
  {  
    "name": "internal.cluster.fqdn",  
    "// additional fields..."  
  }  
]
```

Status codes

Code	Description
200 OK	No error

Updated: November 29, 2021

Users requests

Method	Path	Description
GET	/v1/users	Get all users
GET	/v1/users/{uid}	Get a single user
PUT	/v1/users/{uid}	Update a user's configuration
POST	/v1/users	Create a new user
DELETE	/v1/users/{uid}	Delete a user

Get all users

```
GET /v1/users
```

Get a list of all users.

Permissions

Permission name	Roles
view_all_users_info	admin

Request

Example HTTP request

```
GET /users
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Response

Returns a JSON array of [user objects](#).

Example JSON body

```
[
  {
    "uid": 1,
    "password_issue_date": "2017-03-02T09:43:34Z",
    "email": "user@redislabs.com",
    "name": "John Doe",
    "email_alerts": true,
    "bdb_email_alerts": ["1", "2"],
    "role": "admin",
    "auth_method": "regular"
  },
  {
    "uid": 2,
    "password_issue_date": "2017-03-02T09:43:34Z",
    "email": "user2@redislabs.com",
    "name": "Jane Poe",
    "email_alerts": true,
    "role": "db_viewer",
    "auth_method": "regular"
  }
]
```

Status codes

Code	Description
200 OK	No error

Get user

```
GET /v1/users/{int: uid}
```

Get a single user's details.

Permissions

Permission name	Roles
view_user_info	admin

Request

Example HTTP request

```
GET /users/1
```

Headers

Key	Value	Description
Host	cnn.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The user's unique ID

Response

Returns a [user object](#) that contains the details for the specified user ID.

Example JSON body

```
{  
    "uid": 1,  
    "password_issue_date": "2017-03-07T15:11:08Z",  
    "role": "db_viewer",  
    "email_alerts": true,  
    "bdbs_email_alerts": ["1", "2"],  
    "email": "user@redislabs.com",  
    "name": "John Doe",  
    "auth_method": "regular"  
}
```

Status codes

Code	Description
200 OK	Success.
403 Forbidden	Operation is forbidden.
404 Not Found	User does not exist.

Update user

```
PUT /v1/users/{int: uid}
```

Update an existing user's configuration.

Permissions

Permission name	Roles
update_user	admin

Any user can change their own name, password, or alert preferences.

Request

Example HTTP request

```
PUT /users/1
```

Example JSON body

```
{  
    "name": "Jane Poe",  
    "email_alerts": false  
}
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The user's unique ID

Request body

Include a [user object](#) with updated fields in the request body.

Response

Returns the updated [user object](#).

Example JSON body

```
{
  "uid": 1,
  "password_issue_date": "2017-03-07T15:11:08Z",
  "email": "user@redislabs.com",
  "name": "Jane Poe",
  "email_alerts": false,
  "role": "db_viewer",
  "auth_method": "regular"
}
```

 | Note: For [RBAC-enabled clusters](#), the returned user details include `role_uids` instead of `role`.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information. The following are possible `error_code` values:

Code	Description
password_not_complex	The given password is not complex enough (Only works when the <code>password_complexity</code> feature is enabled).
new_password_same_as_current	The given new password is identical to the old password.
email_already_exists	The given email is already taken.
change_last_admin_role_not_allowed	At least one user with admin role should exist.

Status codes

Code	Description
200 OK	Success, the user is updated.
400 Bad Request	Bad or missing configuration parameters.
404 Not Found	Attempting to change a non-existing user.
406 Not Acceptable	The requested configuration is invalid.

Create user

```
POST /v1/users
```

Create a new user.

Permissions

Permission name	Roles
create_new_user	admin

Request

Example HTTP request

```
POST /users
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Body

Include a single [user object](#) in the request body. The user object must have an email, password, and role.

 | Note: For RBAC-enabled clusters, use `role_uids` instead of `role` in the request body.

`email_alerts` can be configured either as:

- `true` - user will receive alerts for all databases configured in `b dbs_email_alerts`. The user will receive alerts for all databases by default if `b dbs_email_alerts` is not configured. `b dbs_email_alerts` can be a list of database UIDs or ['all'] meaning all databases.
- `false` - user will not receive alerts for any databases

Example JSON body

```
{  
  "email": "newuser@redis.com",  
  "password": "my-password",  
  "name": "Pat Doe",  
  "email_alerts": true,  
  "b dbs_email_alerts": ["1", "2"],  
  "role_uids": [ 3, 4 ],  
  "auth_method": "regular"  
}
```

Response

Returns the newly created [user object](#).

Example JSON body

```
{  
  "uid": 1,  
  "password_issue_date": "2017-03-07T15:11:08Z",  
  "email": "user@redislabs.com",  
  "name": "Jane Poe",  
  "email_alerts": true,  
  "b dbs_email_alerts": ["1", "2"],  
  "role": "db_viewer",  
  "auth_method": "regular"  
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` field that provide additional information.

The following are possible `error_code` values:

Code	Description
<code>password_not_complex</code>	The given password is not complex enough (Only works when the <code>password_complexity</code> feature is enabled).

Code	Description
email_already_exists	The given email is already taken.
name_already_exists	The given name is already taken.

Status codes

Code	Description
200 OK	Success, user is created.
400 Bad Request	Bad or missing configuration parameters.
409 Conflict	User with the same email already exists.

Examples

cURL

```
$ curl -k -X POST -u '[username]:[password]' \
  -H 'Content-Type: application/json' \
  -d '{ "email": "newuser@redis.com", \
    "password": "my-password", \
    "name": "Pat Doe", \
    "email_alerts": true, \
    "bdb_email_alerts": ["1", "2"], \
    "role_uids": [ 3, 4 ], \
    "auth_method": "regular" }' \
  'https://[host][:port]/v1/users'
```

Python

```
import requests
import json

url = "https://[host][:port]/v1/users"
auth = ("[username]", "[password]")

payload = json.dumps({
    "email": "newuser@redis.com",
    "password": "my-password",
    "name": "Pat Doe",
    "email_alerts": True,
    "bdb_email_alerts": [
        "1",
        "2"
    ],
    "role_uids": [
        3,
        4
    ],
    "auth_method": "regular"
})

headers = {
    'Content-Type': 'application/json'
}

response = requests.request("POST", url, auth=auth, headers=headers, data=payload, verify=False)

print(response.text)
```

Delete user

```
DELETE /v1/users/{int: uid}
```

Delete a user.

Permissions

Permission name	Roles
delete_user	admin

Request

Example HTTP request

```
DELETE /users/1
```

Headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

URL parameters

Field	Type	Description
uid	integer	The user's unique ID

Response

Returns a status code to indicate the success or failure of the user deletion.

Status codes

Code	Description
200 OK	Success, the user is deleted.
406 Not Acceptable	The request is not acceptable.

Updated: August 17, 2023

Authorize user requests

Method	Path	Description
POST	/v1/users/authorize	Authorize a user

Authorize user

```
POST /v1/users/authorize
```

Generate a JSON Web Token (JWT) for a user to use as authorization to access the REST API.

Request

Example HTTP request

```
POST /users/authorize
```

Example JSON body

```
{  
  "username": "user@redislabs.com",  
  "password": "my_password"  
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

Include a [JWT authorize object](#) with a valid username and password in the request body.

Response

Returns a JSON object that contains the generated access token.

Example JSON body

```
{  
  "access_token":  
  "eyJ5bGciOiKIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXVi0jE0NjU0NzU0ODYsInVpZFI1IjEiLCJleHAiOjE0NjU0Nz300TZ9.2xYXumd1rDoEi  
}
```

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
password_expired	The password has expired and must be changed.

Status codes

Code	Description
200 OK	The user is authorized.
400 Bad Request	The request could not be understood by the server due to malformed syntax.
401 Unauthorized	The user is unauthorized.

Updated: November 29, 2021

User password requests

Method	Path	Description
--------	------	-------------

Method	Path	Description
PUT	/v1/users/password	Change an existing password
POST	/v1/users/password	Add a new password
DELETE	/v1/users/password	Delete a password

Update password

PUT /v1/users/password

Reset the password list of an internal user to include a new password.

Request

Example HTTP request

PUT /users/password

Example JSON body

```
{
  "username": "johnsmith",
  "old_password": "a password that exists in the current list",
  "new_password": "the new (single) password"
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

The request must contain a single JSON object with the following fields:

Field	Type	Description
username	string	Affected user (required)
old_password	string	A password that exists in the current list (required)
new_password	string	The new password (required)

Response

Returns a status code to indicate password update success or failure.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
password_not_complex	The given password is not complex enough (Only work when the <code>password_complexity</code> feature is enabled).
new_password_same_as_current	The given new password is identical to one of the already existing passwords.

Status codes

Code	Description
200 OK	Success, password changed
400 Bad Request	Bad or missing parameters.
401 Unauthorized	The user is unauthorized.
404 Not Found	Attempting to reset password to a non-existing user.

Add password

POST /v1/users/password

Add a new password to an internal user's passwords list.

Request

Example HTTP request

POST /users/password

Example JSON body

```
{
  "username": "johnsmith",
  "old_password": "an existing password",
  "new_password": "a password to add"
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

The request must contain a single JSON object with the following fields:

Field	Type	Description
username	string	Affected user (required)
old_password	string	A password that exists in the current list (required)
new_password	string	The new (single) password (required)

Response

Returns a status code to indicate password creation success or failure. If an error occurs, the response body may include a more specific error code and message.

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
password_not_complex	The given password is not complex enough (Only work when the <code>password_complexity</code> feature is enabled).
new_password_same_as_current	The given new password is identical to one of the already existing passwords.

Status codes

Code	Description
200 OK	Success, new password was added to the list of valid passwords.
400 Bad Request	Bad or missing parameters.
401 Unauthorized	The user is unauthorized.
404 Not Found	Attempting to add a password to a non-existing user.

Delete password

```
DELETE /v1/users/password
```

Delete a password from an internal user's passwords list.

Request

Example HTTP request

```
DELETE /users/password
```

Example JSON body

```
{
  "username": "johnsmith",
  "old_password": "an existing password"
}
```

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn	Domain name
Accept	application/json	Accepted media type

Request body

The request must contain a single JSON with the following fields:

Field	Type	Description
username	string	Affected user (required)
old_password	string	Existing password to be deleted (required)

Response

Error codes

When errors are reported, the server may return a JSON object with `error_code` and `message` fields that provide additional information. The following are possible `error_code` values:

Code	Description
cannot_delete_last_password	Cannot delete the last password of a user

Status codes

Code	Description
------	-------------

Code	Description
200 OK	Success, new password was deleted from the list of valid passwords.
400 Bad Request	Bad or missing parameters.
401 Unauthorized	The user is unauthorized.
404 Not Found	Attempting to delete a password to a non-existing user.

Updated: November 29, 2021

Refresh JWT requests

Method	Path	Description
POST	/v1/users/refresh_jwt	Get a new authentication token

Get a new authentication token

POST /v1/users/refresh_jwt

Generate a new JSON Web Token (JWT) for authentication.

Takes a valid token and returns the new token generated by the request.

Request

Example HTTP request

POST /users/refresh_jwt

Request headers

Key	Value	Description
Host	cnm.cluster.fqdn JWT eyJ5bGciOiKIUzI1NiIsInR5cCI6IkpxVCJ9.	Domain name
Authorization	eyJpYXViOjE0NjU0NzU0ODYsInVpZFI1IjEiLCJ9. leHAiOjE0NjU0Nz30OTZ9.2xYXumd1rDoE0e dFzcLEIMOHsshaqQk2HUNgdsUKxMU	Valid JSON Web Token (JWT)

Request body

Field	Type	Description
ttl	integer	Time to live - The amount of time in seconds the token will be valid (optional)

Response

Returns a JSON object that contains the generated access token.

Example JSON body

```
{  
    "access_token":  
"eyJ5bGciOiKIUzI1NiIsInR5cCI6IkpxVCJ9.eyJpYXVi0jE0NjU0NzU0ODYsInVpZFI1IjEiLCJleHAiOjE0NjU0Nz300TZ9.2xYXumd1rDoEi  
}
```

Status codes

Code	Description
200 OK	A new token is given.
401 Unauthorized	The token is invalid or password has expired.

Updated: November 29, 2021

Redis Enterprise REST API objects

Certain [REST API requests](#) require you to include specific objects in the request body. Many requests also return objects in the response body.

Both REST API requests and responses represent these objects as [JSON](#).

Object	Description
action	An object that represents cluster actions
alert	An object that contains alert info
bdb	An object that represents a database
bdb_group	An object that represents a group of databases with a shared memory pool
bootstrap	An object for bootstrap configuration
check_result	An object that contains the results of a cluster check
cluster	An object that represents a cluster
cluster_settings	An object for cluster resource management settings
crdb	An object that represents an Active-Active database
crdb_task	An object that represents a CRDB task
db_alerts_settings	An object for database alerts configuration
db_conns_auditing_config	An object for database connection auditing settings
job_scheduler	An object for job scheduler settings
jwt_authorize	An object for user authentication or a JW token refresh request
ldap	An object that contains the cluster's LDAP configuration
ldap_mapping	An object that represents a mapping between an LDAP group and roles
module	An object that represents a Redis module
node	An object that represents a node in the cluster
ocsp	An object that represents the cluster's OCSP configuration
ocsp_status	An object that represents the cluster's OCSP status
proxy	An object that represents a proxy in the cluster
redis_acl	An object that represents a Redis access control list (ACL)
role	An object that represents a role
services_configuration	An object for optional cluster services settings
shard	An object that represents a database shard
state-machine	An object that represents a state machine.
statistics	An object that contains metrics for clusters, databases, nodes, or shards
suffix	An object that represents a DNS suffix

Object	Description
user	An object that represents a Redis Enterprise user

Updated: November 29, 2021

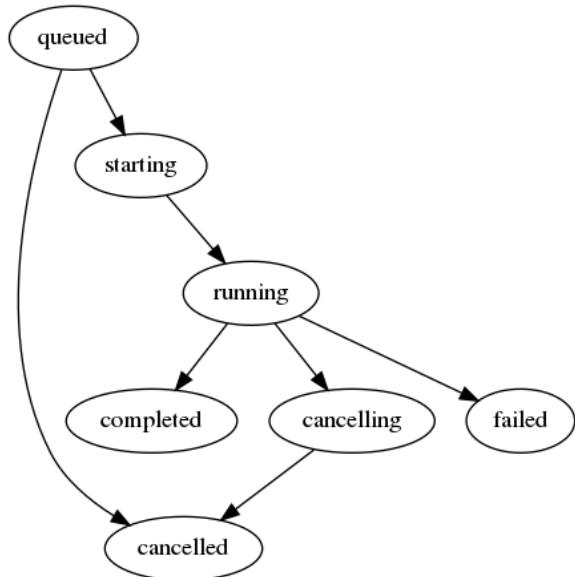
Action object

The cluster allows you to invoke general maintenance actions such as rebalancing or taking a node offline by moving all of its entities to other nodes.

Actions are implemented as tasks in the cluster. Every task has a unique `task_id` assigned by the cluster, a task name which describes the task, a status, and additional task-specific parameters.

The REST API provides a simplified interface that allows callers to invoke actions and query their status without a specific `task_id`.

The action lifecycle is based on the following status and status transitions:



Name	Type/Value	Description
progress	integer (range: 0-100)	Represents percent completed
status	queued starting running cancelling cancelled completed failed	Requested operation and added it to the queue to await processing Picked up operation from the queue and started processing Currently executing operation Operation cancellation is in progress Operation cancelled Operation completed Operation failed

When a task fails, the `error_code` and `error_message` fields describe the error.

Possible `error_code` values:

Code	Description
<code>internal_error</code>	An internal error that cannot be mapped to a more precise error code
<code>insufficient_resources</code>	The cluster does not have sufficient resources to complete the required operation

Alert object

You can view, configure, and enable various alerts for the cluster.

Alerts are bound to a cluster object (such as a [BDB](#) or [node](#)), and the cluster's state determines whether the alerts turn on or off.

Name	Type/Value	Description	Writable
change_time	string	Timestamp when alert state last changed	
change_value	object	Contains data relevant to the evaluation time when the alert went on/off (thresholds, sampled values, etc.)	
enabled	boolean	If true, alert is enabled	x
severity	'DEBUG' 'INFO' 'WARNING' 'ERROR' 'CRITICAL'	The alert's severity	
state	boolean	If true, alert is currently triggered	
threshold	string	Represents an alert threshold when applicable	x

BDB object

An API object that represents a managed database in the cluster.

Name	Type/Value	Description
uid	integer	Cluster unique ID of database. Can be set during creation but cannot be updated.
account_id	integer	SM account ID
action_uid	string	Currently running action's UID (read-only)
aof_policy	'appendfsync-every-sec' 'appendfsync-always'	Policy for Append-Only File data persistence
authentication_admin_pass	string	Password for administrative access to the BDB (used for SYNC from the BDB)
authentication_redis_pass	string	Redis AUTH password authentication. Use for Redis databases only. Ignored for memcached databases. (deprecated as of Redis Enterprise v7.2, replaced with multiple passwords feature in version 6.0.X)
authentication_sasl_pass	string	Binary memcache SASL password
authentication_sasl_uname	string	Binary memcache SASL username (pattern does not allow special characters &,<,>,")

Name	Type/Value	Description
authentication_ssl_client_certs	[{ "client_cert": string }, ...]	List of authorized client certificates client_cert: X.509 PEM (base64) encoded certificate
authentication_ssl_crdt_certs	[{ "client_cert": string }, ...]	List of authorized CRDT certificates client_cert: X.509 PEM (base64) encoded certificate
authorized_names	array of strings	Additional certified names (deprecated as of Redis Enterprise v6.4.2; use authorized_subjects instead)
authorized_subjects	[{ "CN": string, "O": string, "OU": [array of strings], "L": string, "ST": string, "C": string }, ...]	A list of valid subjects used for additional certificate validations during TLS client authentication. All subject attributes are case-sensitive. Required subject fields: "CN" for Common Name Optional subject fields: "O" for Organization "OU" for Organizational Unit (array of strings) "L" for Locality (city) "ST" for State/Province "C" for 2-letter country code
avoid_nodes	array of strings	Cluster node UIDs to avoid when placing the database's shards and binding its endpoints
background_op	[{ "status": string, "name": string, "error": object, "progress": number }, ...]	(read-only); progress: Percent of completed steps in current operation
backup	boolean (default: false)	Policy for periodic database backup
backup_failure_reason	'no-permission' 'wrong-file-path' 'general-error'	Reason of last failed backup process (read-only)
backup_history	integer (default: 0)	Backup history retention policy (number of days, 0 is forever)
backup_interval	integer	Interval in seconds in which automatic backup will be initiated
backup_interval_offset	integer	Offset (in seconds) from round backup interval when automatic backup will be initiated (should be less than backup_interval)
backup_location	complex object	Target for automatic database backups. Call GET / jsonschema to retrieve the object's structure.
backup_progress	number, (range: 0-100)	Database scheduled periodic backup progress (percentage) (read-only)
backup_status	'exporting' 'succeeded' 'failed'	Status of scheduled periodic backup process (read-only)
bigstore	boolean (default: false)	Database bigstore option
bigstore_ram_size	integer (default: 0)	Memory size of bigstore RAM part.

Name	Type/Value	Description
bigstore_ram_weights	[{ "shard_uid": integer, "weight": number }, ...]	List of shard UIDs and their bigstore RAM weights; shard_uid : Shard UID; weight : Relative weight of RAM distribution
client_cert_subject_validation_type	disabled san_cn full_subject	Enables additional certificate validations that further limit connections to clients with valid certificates during TLS client authentication. disabled : Authenticates clients with valid certificates. No additional validations are enforced. san_cn : A client certificate is valid only if its Common Name (CN) matches an entry in the list of valid subjects. Ignores other Subject attributes. full_subject : A client certificate is valid only if its Subject attributes match an entry in the list of valid subjects.
crdt	boolean (default: false)	Use CRDT-based data types for multi-master replication
crdt_causal_consistency	boolean (default: false)	Causal consistent CRDB.
crdt_config_version	integer	Replica-set configuration version, for internal use only.
crdt_featureset_version	integer	CRDB active FeatureSet version
crdt_ghost_replica_ids	string	Removed replicas IDs, for internal use only.
crdt_guid	string	GUID of CRDB this database belongs to, for internal use only.
crdt_protocol_version	integer	CRDB active Protocol version
crdt_repl_backlog_size	string	Active-Active replication backlog size ('auto' or size in bytes)
crdt_replica_id	integer	Local replica ID, for internal use only.
crdt_replicas	string	Replica set configuration, for internal use only.
crdt_sources	array of syncer_sources objects	Remote endpoints/peers of CRDB database to sync from. See the 'bdb -> replica_sources' section
crdt_sync	'enabled' 'disabled' 'paused' 'stopped'	Enable, disable, or pause syncing from specified crdt_sources. Applicable only for Active-Active databases. See replica_sync for more details.
crdt_sync_dist	boolean	Enable/disable distributed syncer in master-master
crdt_syncer_auto_oom_unlatch	boolean (default: true)	Syncer automatically attempts to recover synchronisation from peers after this database throws an Out-Of-Memory error. Otherwise, the syncer exits
created_time	string	The date and time the database was created (read-only)
data_internode_encryption	boolean	Should the data plane internode communication for this database be encrypted
data_persistence	'disabled' 'snapshot' 'aof'	Database on-disk persistence policy. For snapshot persistence, a snapshot_policy must be provided
dataset_import_sources	complex object	Array of source file location description objects to import from when performing an import action. This is write-only and cannot be read after set. Call GET /jsonschema to retrieve the object's structure.
db_conns_auditing	boolean	Enables/deactivates database connection auditing
default_user	boolean (default: true)	Allow/disallow a default user to connect
disabled_commands	string (default:)	Redis commands which are disabled in db

Name	Type/Value	Description
dns_address_master	string	Database private address endpoint FQDN (read-only) (deprecated as of Redis Enterprise v4.3.3)
email_alerts	boolean (default: false)	Send email alerts for this DB
endpoint	string	Latest bound endpoint. Used when reconfiguring an endpoint via update
endpoint_ip	complex object	External IP addresses of node hosting the BDB's endpoint. GET /j sonschema to retrieve the object's structure. (read-only) (deprecated as of Redis Enterprise v4.3.3)
endpoint_node	integer	Node UID hosting the BDB's endpoint (read-only) (deprecated as of Redis Enterprise v4.3.3)
endpoints	array	List of database access endpoints (read-only)
enforce_client_authentication	'enabled' 'disabled'	Require authentication of client certificates for SSL connections to the database. If set to 'enabled', a certificate should be provided in either authentication_ssl_client_certs or authentication_ssl_crft_certs
eviction_policy	'volatile-lru' 'volatile-ttl' 'volatile-random' 'allkeys-lru' 'allkeys-random' 'noeviction' 'volatile-lfu' 'allkeys-lfu'	Database eviction policy (Redis style). Redis DB default: 'volatile-lru' memcached DB default: 'allkeys-lru'
export_failure_reason	'no-permission' 'wrong-file-path' 'general-error'	Reason of last failed export process (read-only)
export_progress	number, (range: 0-100)	Database manually triggered export progress (percentage) (read-only)
export_status	'exporting' 'succeeded' 'failed'	Status of manually triggered export process (read-only)
generate_text_monitor	boolean	Enable/disable generation of syncer monitoring information
gradual_src_max_sources	integer (default: 1)	Sync a maximum N sources in parallel (gradual_src_mode should be enabled for this to take effect)
gradual_src_mode	'enabled' 'disabled'	Indicates if gradual sync (of sync sources) should be activated
gradual_sync_max_shards_per_source	integer (default: 1)	Sync a maximum of N shards per source in parallel (gradual_sync_mode should be enabled for this to take effect)
gradual_sync_mode	'enabled' 'disabled' 'auto'	Indicates if gradual sync (of source shards) should be activated ('auto' for automatic decision)
hash_slots_policy	'legacy' '16k'	The policy used for hash slots handling 'legacy': slots range is '1-4096' '16k': slots range is '0-16383'
implicit_shard_key	boolean (default: false)	Controls the behavior of what happens in case a key does not match any of the regex rules. true: if a key does not match any of the rules, the entire key will be used for the hashing function false: if a key does not match any of the rules, an error will be returned.

Name	Type/Value	Description
import_failure_reason	'download-error' 'file-corrupted' 'general-error' 'file-larger-than-mem-limit:<n bytes of expected dataset>,<n bytes configured bdb limit>' 'key-too-long' 'invalid-bulk-length' 'out-of-memory'	Import failure reason (read-only)
import_progress	number, (range: 0-100)	Database import progress (percentage) (read-only)
import_status	'idle' 'initializing' 'importing' 'succeeded' 'failed'	Database import process status (read-only)
internal	boolean (default: false)	Is this a database used by the cluster internally
last_backup_time	string	Time of last successful backup (read-only)
last_changed_time	string	Last administrative configuration change (read-only)
last_export_time	string	Time of last successful export (read-only)
max_aof_file_size	integer	Maximum size for shard's AOF file (bytes). Default 300GB, (on bigstore DB 150GB)
max_aof_load_time	integer (default: 3600)	Maximum time shard's AOF reload should take (seconds).
max_connections	integer (default: 0)	Maximum number of client connections allowed (0 unlimited)
memory_size	integer (default: 0)	Database memory limit (0 is unlimited), expressed in bytes.
metrics_export_all	boolean	Enable/disable exposing all shard metrics through the metrics exporter
mkms	boolean (default: true)	Are MKMS (Multi Key Multi Slots) commands supported?
module_list	[{ "module_id": string, "module_args": [u'string', u'null'], "module_name": string, "semantic_version": string }, ...]	List of modules associated with database module_id: Module UID module_args: Module command line arguments (pattern does not allow special characters &,<,>,") module_name: Module's name semantic_version: Module's semantic version
mtls_allow_outdated_certs	boolean	An optional mTLS relaxation flag for certs verification
mtls_allow_weak_hashing	boolean	An optional mTLS relaxation flag for certs verification
name	string	Database name
oss_cluster	boolean (default: false)	OSS Cluster mode option. Cannot be enabled with 'hash_slots_policy': 'legacy'
oss_cluster_api_preferred_ip_type	'internal' 'external'	Internal/external IP type in OSS cluster API. Default value for new endpoints
oss_sharding	boolean (default: false)	An alternative to <code>shard_key_regex</code> for using the common case of the OSS shard hashing policy
port	integer	TCP port on which the database is available. Generated automatically if omitted and returned as 0
proxy_policy	'single' 'all-master-shards' 'all-nodes'	The default policy used for proxy binding to endpoints
rack_aware	boolean (default: false)	Require the database to always replicate across multiple racks

Name	Type/Value	Description
redis_version	string	Version of the redis-server processes: e.g. 6.0, 5.0-big
repl_backlog_size	string	Redis replication backlog size ('auto' or size in bytes)
replica_sources	array of syncer_sources objects	Remote endpoints of database to sync from. See the 'bdb -> replica_sources' section
replica_sync	'enabled' 'disabled' 'paused' 'stopped'	Enable, disable, or pause syncing from specified replica_sources
replica_sync_dist	boolean	Enable/disable distributed syncer in replica-of
replication	boolean (default: false)	In-memory database replication mode
resp3	boolean (default: true)	Enables or deactivates RESP3 support
roles_permissions	[{ "role_uid": integer, "redis_acl_uid": integer }, ...]	
shard_block_crossslot_keys	boolean (default: false)	In Lua scripts, prevent use of keys from different hash slots within the range owned by the current shard
shard_block_foreign_keys	boolean (default: true)	In Lua scripts, foreign_keys prevent use of keys which could reside in a different shard (foreign keys)
shard_key_regex	[{ "regex": string }, ...]	To use the default rules you should set the value to: Custom keyname-based sharding rules.
	[{ "regex": ".*\\{(?:< tag >.*)\\}.*" }, { "regex": "(?:< tag >.*)" }]	
shard_list	array of integers	Cluster unique IDs of all database shards.
sharding	boolean (default: false)	Cluster mode (server-side sharding). When true, shard hashing rules must be provided by either oss_sharding or shard_key_regex
shards_count	integer, (range: 1-512) (default: 1)	Number of database server-side shards
shards_placement	'dense' 'sparse'	Control the density of shards 'dense': Shards reside on as few nodes as possible 'sparse': Shards reside on as many nodes as possible
skip_import_analyze	'enabled' 'disabled'	Enable/disable skipping the analysis stage when importing an RDB file
slave_buffer	'auto' value in MB hard:soft:time	Redis replica output buffer limits
slave_ha	boolean	Enable replica high availability mechanism for this database (default takes the cluster setting)
slave_ha_priority	integer	Priority of the BDB in replica high availability mechanism

Name	Type/Value	Description
snapshot_policy	array of snapshot_policy objects	Policy for snapshot-based data persistence. A dataset snapshot will be taken every N secs if there are at least M writes changes in the dataset
ssl	boolean (default: false)	Require SSL authenticated and encrypted connections to the database (deprecated as of Redis Enterprise v5.0.1)
status	'pending' 'active' 'active-change-pending' 'delete-pending' 'import-pending' 'creation-failed' 'recovery' 'enabled' 'disabled' 'paused' 'stopped'	Database lifecycle status (read-only)
sync		(deprecated as of Redis Enterprise v5.0.1, use replica_sync or crdt_sync instead) Enable, disable, or pause syncing from specified sync_sources
sync_sources	[{ "uid": integer, "uri": string, "compression": integer, "status": string, "rdb_transferred": integer, "rdb_size": integer, "last_update": string, "lag": integer, "last_error": string }, ...]	(deprecated as of Redis Enterprise v5.0.1, instead use replica_sources or crdt_sources) Remote endpoints of database to sync from. See the 'bdb -> replica_sources' section uid : Numeric unique identification of this source uri : Source Redis URI compression : Compression level for the replication link status : Sync status of this source rdb_transferred : Number of bytes transferred from the source's RDB during the syncing phase rdb_size : The source's RDB size to be transferred during the syncing phase last_update : Time last update was received from the source lag : Lag in millisec between source and destination (while synced) last_error : Last error encountered when syncing from the source
syncer_mode	'distributed' 'centralized'	The syncer for replication between database instances is either on a single node (centralized) or on each node that has a proxy according to the proxy policy (distributed). (read-only)
tags	[{ "key": string, "value": string }, ...]	Optional list of tags objects attached to the database key : Represents the tag's meaning and must be unique among tags (pattern does not allow special characters &,<,>,") value : The tag's value
tls_mode	'enabled' 'disabled' 'replica_ssl'	Require TLS-authenticated and encrypted connections to the database
type	'redis' 'memcached'	Type of database
use_nodes	array of strings	Cluster node UIDs to use for database shards and bound endpoints
version	string	Database compatibility version: full Redis/memcached version number, such as 6.0.6. This value can only change during database creation and database upgrades.
wait_command	boolean (default: true)	Supports Redis wait command (read-only)

Updated: August 25, 2023

BDB backup/export location object

You can back up or export a database's dataset to the following types of locations:

- FTP/S
- SFTP
- Amazon S3
- Google Cloud Storage
- Microsoft Azure Storage
- NAS/Local Storage

Basic parameters

For all backup/export location objects, you need to specify the location type via the `type` field.

Location type	"type" value
FTP/S	"url"
SFTP	"sftp"
Amazon S3	"s3"
Google Cloud Storage	"gs"
Microsoft Azure Storage	"abs"
NAS/Local Storage	"mount_point"

Location-specific parameters

Any additional required parameters may differ based on the backup/export location type.

FTP

Key name	Type	Description
url	string	A URI that represents a FTP/S location with the following format: <code>ftp://user:password@host:port/path/</code> . The user and password can be omitted if not needed.

SFTP

Key name	Type	Description
key	string	SSH private key to secure the SFTP server connection. If you do not specify an SSH private key, the autogenerated private key of the cluster is used, and you must add the SSH public key of the cluster to the SFTP server configuration. (optional)
sftp_url	string	SFTP URL in the format: <code>sftp://user:password@host[:port]/[path/]</code> . The default port number is 22 and the default path is '/'.

AWS S3

Key name	Type	Description
access_key_id	string	The AWS Access Key ID with access to the bucket
bucket_name	string	S3 bucket name
secret_access_key	string	The AWS Secret Access Key that matches the Access Key ID

Key name	Type	Description
subdir	string	Path to the backup directory in the S3 bucket (optional)

Google Cloud Storage

Key name	Type	Description
bucket_name	string	Cloud Storage bucket name
client_email	string	Email address for the Cloud Storage client ID
client_id	string	Cloud Storage client ID with access to the Cloud Storage bucket
private_key	string	Cloud Storage private key that matches the private key ID
private_key_id	string	Cloud Storage private key ID with access to the Cloud Storage bucket
subdir	string	Path to the backup directory in the Cloud Storage bucket (optional)

Azure Blob Storage

Key name	Type	Description
account_key	string	Access key for the storage account
account_name	string	Storage account name with access to the container
container	string	Blob Storage container name
sas_token	string	Token to authenticate with shared access signature
subdir	string	Path to the backup directory in the Blob Storage container (optional)



Note: account_key and sas_token are mutually exclusive

NAS/Local Storage

Key name	Type	Description
path	string	Path to the local mount point. You must create the mount point on all nodes, and the redislabs:redislabs user must have read and write permissions on the local mount point.

Updated: December 9, 2021

BDB dataset import sources object

You can import data to a database from the following location types:

- HTTP/S
- FTP
- SFTP
- Amazon S3
- Google Cloud Storage
- Microsoft Azure Storage
- NAS/Local Storage

The source file to import should be in the [RDB](#) format. It can also be in a compressed (.gz) RDB file.

Supply an array of dataset import source objects to import data from multiple files.

Basic parameters

For all import location objects, you need to specify the location type via the `type` field.

Location type	"type" value
FTP/S	"url"
SFTP	"sftp"
Amazon S3	"s3"
Google Cloud Storage	"gs"
Microsoft Azure Storage	"abs"
NAS/Local Storage	"mount_point"

Location-specific parameters

Any additional required parameters may differ based on the import location type.

FTP

Key name	Type	Description
url	string	A URI that represents the FTP/S location with the following format: <code>ftp://user:password@host:port/path/</code> . The user and password can be omitted if not needed.

SFTP

Key name	Type	Description
key	string	SSH private key to secure the SFTP server connection. If you do not specify an SSH private key, the autogenerated private key of the cluster is used and you must add the SSH public key of the cluster to the SFTP server configuration. (optional)
sftp_url	string	SFTP URL in the format: <code>sftp://user:password@host[:port]/path/</code> . The default port number is 22 and the default path is '/'.

AWS S3

Key name	Type	Description
access_key_id	string	The AWS Access Key ID with access to the bucket
bucket_name	string	S3 bucket name
secret_access_key	string	The AWS Secret Access that matches the Access Key ID
subdir	string	Path to the backup directory in the S3 bucket (optional)

Google Cloud Storage

Key name	Type	Description
bucket_name	string	Cloud Storage bucket name
client_email	string	Email address for the Cloud Storage client ID

Key name	Type	Description
client_id	string	Cloud Storage client ID with access to the Cloud Storage bucket
private_key	string	Private key for the Cloud Storage matching the private key ID
private_key_id	string	Cloud Storage private key ID with access to the Cloud Storage bucket
subdir	string	Path to the backup directory in the Cloud Storage bucket (optional)

Azure Blob Storage

Key name	Type	Description
account_key	string	Access key for the storage account
account_name	string	Storage account name with access to the container
container	string	Blob Storage container name
sas_token	string	Token to authenticate with shared access signature
subdir	string	Path to the backup directory in the Blob Storage container (optional)



Note: account_key and sas_token are mutually exclusive

NAS/Local Storage

Key name	Type	Description
path	string	Path to the locally mounted filename to import. You must create the mount point on all nodes, and the redislabs:redislabs user must have read permissions on the local mount point.

Updated: April 14, 2022

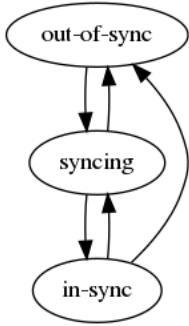
BDB replica sources status field

The `replica_sources` status field relates to the [Replica Of](#) feature, which enables the creation of a Redis database (single- or multi-shard) that synchronizes data from another Redis database (single- or multi-shard).

The status field represents the Replica Of sync status for a specific sync source.

Possible status values:

Status	Description	Possible next status
'out-of-sync'	Sync process is disconnected from source and trying to reconnect	'syncing'
'syncing'	Sync process is in progress	'in-sync' 'out-of-sync'
'in-sync'	Sync process finished successfully, and new commands are syncing on a regular basis	'syncing' 'out-of-sync'



Updated: August 31, 2022

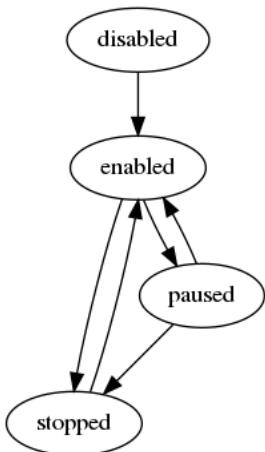
BDB replica sync field

The BDB `replica_sync` field relates to the [Replica Of](#) feature, which enables the creation of a Redis database (single- or multi-shard) that synchronizes data from another Redis database (single- or multi-shard).

You can use the `replica_sync` field to enable, disable, or pause the [Replica Of](#) sync process. The BDB `crdt_sync` field has a similar purpose for the Redis CRDB.

Possible BDB sync values:

Status	Description	Possible next status
'disabled'	(default value) Disables the sync process and represents that no sync is currently configured or running.	'enabled'
'enabled'	Enables the sync process and represents that the process is currently active.	'stopped' 'paused'
'paused'	Pauses the sync process. The process is configured but is not currently executing any sync commands.	'enabled' 'stopped'
'stopped'	An unrecoverable error occurred during the sync process, which caused the system to stop the sync.	'enabled'



When the sync is in the 'stopped' or 'paused' state, then the `last_error` field in the relevant source entry in the `sync_sources` "status" field contains the detailed error message.

Updated: August 31, 2022

Snapshot policy object

Name	Type/Value	Description
secs	integer	Interval in seconds between snapshots
writes	integer	Number of write changes required to trigger a snapshot

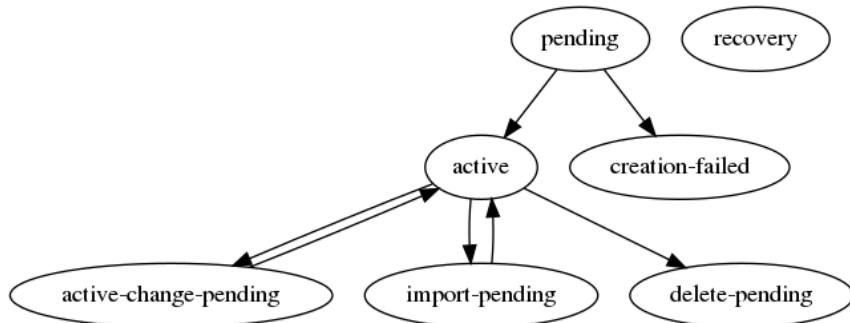
Updated: November 29, 2021

BDB status field

The BDB status field is a read-only field that represents the database status.

Possible status values:

Status	Description	Possible next status
'active'	Database is active and no special action is in progress	'active-change-pending' 'import-pending' 'delete-pending'
'active-change-pending'		'active'
'creation-failed'	Initial database creation failed	
'delete-pending'	Database deletion is in progress	
'import-pending'	Dataset import is in progress	'active'
'pending'	Temporary status during database creation	'active' 'creation-failed'
'recovery'	Not currently relevant (intended for future use)	



Updated: November 29, 2021

Syncer sources object

Name	Type/Value	Description
uid	integer	Unique ID of this source
client_cert	string	Client certificate to use if encryption is enabled
client_key	string	Client key to use if encryption is enabled
compression	integer, (range: 0-6)	Compression level for the replication link
encryption	boolean	Encryption enabled/disabled
lag	integer	Lag in milliseconds between source and destination (while synced)
last_error	string	Last error encountered when syncing from the source

Name	Type/Value	Description
last_update	string	Time when we last received an update from the source
rdbsize	integer	The source's RDB size to be transferred during the syncing phase
rdb_transferred	integer	Number of bytes transferred from the source's RDB during the syncing phase
replication_tls_sni	string	Replication TLS server name indication
server_cert	string	Server certificate to use if encryption is enabled
status	string	Sync status of this source
uri	string	Source Redis URI

Updated: November 29, 2021

BDB group object

An API object that represents a group of databases that share a memory pool.

Name	Type/Value	Description
uid	integer	Cluster unique ID of the database group
members	array of strings	A list of UIDs of member databases (read-only)
memory_size	integer	The common memory pool size limit for all databases in the group, expressed in bytes

Updated: November 29, 2021

Bootstrap object

A bootstrap configuration object.

Name	Type/Value	Description
action	'create_cluster' 'join_cluster' 'recover_cluster'	Action to perform
cluster	cluster_identity object	Cluster to join or create
cnm_https_port	integer	Port to join a cluster with non-default cnm_https port
credentials	credentials object	Cluster admin credentials
dns_suffixes	[{ "name": string, "cluster_default": boolean, "use_aaaa_ns": boolean, "use_internal_addr": boolean, "slaves": array }, ...]	Explicit configuration of DNS suffixes name: DNS suffix name cluster_default: Should this suffix be the default cluster suffix use_aaaa_ns: Should AAAA records be published for NS records use_internal_addr: Should internal cluster IPs be published for databases slaves: List of replica servers that should be published as NS and notified
license	string	License string
max_retries	integer	Max number of retries in case of recoverable errors
node	node_identity object	Node description

Name	Type/Value	Description
policy	policy object	Policy object
recovery_filename	string	Name of backup file to recover from
required_version	string	This node can only join the cluster if all nodes in the cluster have a version greater than the required_version
retry_time	integer	Max waiting time between retries (in seconds)

Updated: November 29, 2021

Cluster identity object

Name	Type/Value	Description
name	string	Fully qualified cluster name. Limited to 64 characters and must comply with the IETF's RFC 952 standard and section 2.1 of the RFC 1123 standard.
nodes	array of strings	Array of IP addresses of existing cluster nodes
wait_command	boolean (default: true)	Supports Redis wait command

Updated: November 29, 2021

Credentials object

Name	Type/Value	Description
password	string	Admin password
username	string	Admin username (pattern does not allow special characters &,<,>,")

Updated: November 29, 2021

Identity object

Name	Type/Value	Description
uid	integer	Assumed node's UID to join cluster. Used to replace a dead node with a new one.
accept_servers	boolean (default: true)	If true, no shards will be created on the node
addr	string	Internal IP address of node
external_addr	complex object	External IP addresses of node. GET /jsonschema to retrieve the object's structure.
name	string	Node's name
override_rack_id	boolean	When replacing an existing node in a rack-aware cluster, allows the new node to be located in a different rack
rack_id	string	Rack ID, overrides cloud config

Updated: November 29, 2021

Limits object

Name	Type/Value	Description
max_listeners	integer (default: 100)	Max allowed listeners on node
max_redis_servers	integer (default: 100)	Max allowed Redis servers on node

Updated: November 29, 2021

Node identity object

Name	Type/Value	Description
bigstore_driver	'ibm-capi-ga1' 'ibm-capi-ga2' 'ibm-capi-ga4' 'speedb' 'rocksdb'	Bigstore driver name or none
identity	identity object	Node identity
limits	limits object	Node limits
paths	paths object	Storage paths object

Updated: August 17, 2023

Paths object

Name	Type/Value	Description
bigstore_path	string	Bigredis storage path
ccs_persistent_path	string	Persistent storage path of CCS
ephemeral_path	string	Ephemeral storage path
persistent_path	string	Persistent storage path

Updated: November 29, 2021

Policy object

Name	Type/Value	Description
default_fork_evict_ram	boolean (default: false)	If true, the databases should evict data from RAM to ensure successful replication or persistence
default_non_sharded_proxy_policy	'single' 'all-master-shards' 'all-nodes'	Default proxy_policy for newly created non-sharded databases' endpoints
default_sharded_proxy_policy	'single' 'all-master-shards' 'all-nodes'	Default proxy_policy for newly created sharded databases' endpoints

Name	Type/Value	Description
default_shards_placement	'dense' 'sparse'	Default shards_placement for newly created databases
rack_aware	boolean	Cluster rack awareness
shards_overbooking	boolean (default: true)	If true, all databases' memory_size settings are ignored during shards placement

Updated: November 29, 2021

Check result object

Cluster check result

Name	Type/Value	Description
cluster_test_result	boolean	Indication if any of the tests failed
nodes	[{ "node_uid": integer, "result": boolean, "error": string }, ...]	Nodes results

Updated: November 29, 2021

Cluster object

An API object that represents the cluster.

Name	Type/Value	Description
alert_settings	alert_settings object	Cluster and node alert settings
bigstore_driver	'speedb' 'rocksdb'	Storage engine for Auto Tiering
cluster_ssh_public_key	string	Cluster's autogenerated SSH public key
cm_port	integer, (range: 1024-65535)	UI HTTPS listening port
cm_session_timeout_minutes	integer (default: 15)	The timeout (in minutes) for the session to the CM
cnm_http_port	integer, (range: 1024-65535)	API HTTP listening port
cnm_https_port	integer, (range: 1024-65535)	API HTTPS listening port
control_cipher_suites	string	Specifies the enabled ciphers for the control plane. The ciphers are specified in the format understood by the BoringSSL library.
crdt_rest_client_retries	integer	Maximum number of retries for the REST client used by the Active-Active management API
crdt_rest_client_timeout	integer	Timeout for REST client used by the Active-Active management API
created_time	string	Cluster creation date (read-only)
data_cipher_list	string	Specifies the enabled ciphers for the data plane. The ciphers are specified in the format understood by the OpenSSL library.
debuginfo_path	string	Path to a local directory used when generating support packages

Name	Type/Value	Description
default_non_sharded_proxy_policy	string (default: single)	Default proxy_policy for newly created non-sharded databases' endpoints (read-only)
default_sharded_proxy_policy	string (default: all-master-shards)	Default proxy_policy for newly created sharded databases' endpoints (read-only)
email_alerts	boolean (default: false)	Send node/cluster email alerts (requires valid SMTP and email_from settings)
email_from	string	Sender email for automated emails
encrypt_pkeys	boolean (default: false)	Enable or turn off encryption of private keys
envoy_admin_port	integer, (range: 1024-65535)	Envoy admin port. Changing this port during runtime might result in an empty response because envoy serves as the cluster gateway.
envoy_max_downstream_connections	integer, (range: 100-2048)	The max downstream connections envoy is allowed to open
envoy_mgmt_server_port	integer, (range: 1024-65535)	Envoy management server port
gossip_envoy_admin_port	integer, (range: 1024-65535)	Gossip envoy admin port
handle_redirects	boolean (default: false)	Handle API HTTPS requests and redirect to the master node internally
http_support	boolean (default: false)	Enable or turn off HTTP support
min_control_TLS_version	'1' '1.1' '1.2' '1.3'	The minimum version of TLS protocol which is supported at the control path
min_data_TLS_version	'1' '1.1' '1.2'	The minimum version of TLS protocol which is supported at the data path
min_sentinel_TLS_version	'1' '1.1' '1.2'	The minimum version of TLS protocol which is supported at the data path
name	string	Cluster's fully qualified domain name (read-only)
password_complexity	boolean (default: false)	Enforce password complexity policy
password_expiration_duration	integer (default: 0)	The number of days a password is valid until the user is required to replace it
proxy_certificate	string	Cluster's proxy certificate
proxy_max_ccs_disconnection_time	integer	Cluster-wide proxy timeout policy between proxy and CCS
rack_aware	boolean	Cluster operates in a rack-aware mode (read-only)
s3_url	string	Specifies the URL for S3 export and import
saslauthd_ldap_conf	string	saslauthd LDAP configuration
sentinel_cipher_suites	array	Specifies the list of enabled ciphers for the sentinel service. The supported ciphers are ones that were implemented by the cipher_suits.go package.
sentinel_ssl_policy	'allowed' 'disabled' 'required'	Determines whether the discovery service allows, blocks, or requires TLS connections allowed: Allows both TLS and non-TLS connections disabled: Allows only non-TLS connections required: Allows only TLS connections
slave_ha	boolean (default: false)	Enable the replica high-availability mechanism (read-only)
slave_ha_bdb_coldown_period	integer (default: 86400)	Time in seconds between runs of the replica high-availability mechanism on different nodes on the same database (read-only)
slave_ha_coldown_period	integer (default: 3600)	Time in seconds between runs of the replica high-availability mechanism on different nodes (read-only)

Name	Type/Value	Description
slave_ha_grace_period	integer (default: 900)	Time in seconds between a node failure and when the replica high-availability mechanism starts relocating shards (read-only)
slowlog_in_sanitized_support	boolean	Whether to include slowlogs in the sanitized support package
smtp_host	string	SMTP server for automated emails
smtp_password	string	SMTP server password
smtp_port	integer	SMTP server port for automated emails
smtp_tls_mode	'none' 'starttls' 'tls'	Specifies which TLS mode to use for SMTP access
smtp_use_tls	boolean (default: false)	Use TLS for SMTP access (deprecated as of Redis Enterprise v4.3.3, use smtp_tls_mode field instead)
smtp_username	string	SMTP server username (pattern does not allow special characters &,<,>,")
syncer_certificate	string	Cluster's syncer certificate
upgrade_mode	boolean (default: false)	Is cluster currently in upgrade mode
use_external_ipv6	boolean (default: true)	Should redislabs services listen on ipv6
use_ipv6	boolean (default: true)	Should redislabs services listen on ipv6 (deprecated as of Redis Enterprise v6.4.2, replaced with use_external_ipv6)
wait_command	boolean (default: true)	Supports Redis wait command (read-only)

Updated: August 17, 2023

Alert settings object

Name	Type/Value	Description
cluster_certs_about_to_expire	cluster_alert_settings_with_threshold object	Cluster certificate will expire in x days
cluster_even_node_count	boolean (default: false)	True high availability requires an odd number of nodes in the cluster
cluster_flash_overcommit	boolean (default: false)	Flash memory committed to databases is larger than cluster total flash memory
cluster_inconsistent_redis_sw	boolean (default: false)	Some shards in the cluster are running different versions of Redis software
cluster_inconsistent_rl_sw	boolean (default: false)	Some nodes in the cluster are running different versions of Redis Enterprise software
cluster_internal_bdb	boolean (default: false)	Issues with internal cluster databases
cluster_multiple_nodes_down	boolean (default: false)	Multiple cluster nodes are down (this might cause data loss)
cluster_node_joined	boolean (default: false)	New node joined the cluster
cluster_node_remove_abort_completed	boolean (default: false)	Cancel node remove operation completed
cluster_node_remove_abort_failed	boolean (default: false)	Cancel node remove operation failed
cluster_node_remove_completed	boolean (default: false)	Node removed from the cluster
cluster_node_remove_failed	boolean (default: false)	Failed to remove a node from the cluster
cluster_ocsp_query_failed	boolean (default: false)	Failed to query the OCSP server
cluster_ocsp_status_revoked	boolean (default: false)	OCSP certificate status is REVOKED
cluster_ram_overcommit	boolean (default: false)	RAM committed to databases is larger than cluster total RAM
cluster_too_few_nodes_for_replication	boolean (default: false)	Replication requires at least 2 nodes in the cluster
node_aof_slow_disk_io	boolean (default: false)	AOF reaching disk I/O limits

Name	Type/Value	Description
node_checks_error	boolean (default: false)	Some node checks have failed
node_cpu_utilization	cluster_alert_settings_with_threshold object	Node CPU utilization has reached the threshold value (% of the utilization limit)
node_ephemeral_storage	cluster_alert_settings_with_threshold object	Node ephemeral storage has reached the threshold value (% of the storage limit)
node_failed	boolean (default: false)	Node failed
node_free_flash	cluster_alert_settings_with_threshold object	Node flash storage has reached the threshold value (% of the storage limit)
node_insufficient_disk_aofrw	boolean (default: false)	Insufficient AOF disk space
node_internal_certs_about_to_expire	cluster_alert_settings_with_threshold object	Internal certificate on node will expire in x days
node_memory	cluster_alert_settings_with_threshold object	Node memory has reached the threshold value (% of the memory limit)
node_net_throughput	cluster_alert_settings_with_threshold object	Node network throughput has reached the threshold value (bytes/s)
node_persistent_storage	cluster_alert_settings_with_threshold object	Node persistent storage has reached the threshold value (% of the storage limit)

Updated: August 10, 2022

Cluster alert settings with threshold object

Name	Type/Value	Description
enabled	boolean (default: false)	Alert enabled or disabled
threshold	string	Threshold for alert going on/off

Updated: November 29, 2021

Cluster settings object

Cluster resources management policy

Name	Type/Value	Description
acl_pubsub_default	resetchannels allchannels	Default pub/sub ACL rule for all databases in the cluster: <ul style="list-style-type: none"> • <code>resetchannels</code> blocks access to all channels (restrictive) • <code>allchannels</code> allows access to all channels (permissive)
bigstore_migrate_node_threshold	integer	Minimum free memory (excluding reserved memory) allowed on a node before automatic migration of shards from it to free more memory
bigstore_migrate_node_threshold_p	integer	Minimum free memory (excluding reserved memory) allowed on a node before automatic migration of shards from it to free more memory
bigstore_provision_node_threshold	integer	Minimum free memory (excluding reserved memory) allowed on a node before new shards can no longer be added to it

Name	Type/Value	Description
bigstore_provision_node_threshold_p	integer	Minimum free memory (excluding reserved memory) allowed on a node before new shards can no longer be added to it
data_internode_encryption	boolean	Enable/deactivate encryption of the data plane internode communication
db_conns_auditing	boolean	Audit connections for new databases by default if set to true.
default_concurrent_restore_actions	integer	Default number of restore actions allowed at the same time. Set to 0 to allow any number of simultaneous restore actions.
default_fork_evict_ram	boolean	If true, the b dbs should evict data from RAM to ensure successful replication or persistence
default_non_sharded_proxy_policy	single all-master-shards all-nodes	Default proxy_policy for newly created non-sharded databases' endpoints
default_provisioned_redis_version	string	Default Redis version
default_sharded_proxy_policy	single all-master-shards all-nodes	Default proxy_policy for newly created sharded databases' endpoints
default_shards_placement	dense sparse	Default shards_placement for a newly created databases
endpoint_rebind_propagation_grace_time	integer	Time to wait between the addition and removal of a proxy
failure_detection_sensitivity	high low	Predefined thresholds and timeouts for failure detection (previously known as <code>watchdog_profile</code>) <ul style="list-style-type: none"> • high (previously local-network) – high failure detection sensitivity, lower thresholds, faster failure detection and failover • low (previously cloud) – low failure detection sensitivity, higher tolerance for latency variance (also called network jitter)
login_lockout_counter_reset_after	integer	Number of seconds that must elapse between failed sign in attempts before the lockout counter is reset to 0.
login_lockout_duration	integer	Duration (in secs) of account lockout. If set to 0, the account lockout will persist until released by an admin.
login_lockout_threshold	integer	Number of failed sign in attempts allowed before locking a user account
max_saved_events_per_type	integer	Maximum saved events per event type
max_simultaneous_backups	integer (default: 4)	Maximum number of backup processes allowed at the same time
parallel_shards_upgrade	integer	Maximum number of shards to upgrade in parallel
persistence_cleanup_scan_interval	string	CRON expression that defines the Redis cleanup schedule
rack_aware	boolean	Cluster operates in a rack-aware mode
redis_migrate_node_threshold	integer	Minimum free memory (excluding reserved memory) allowed on a node before automatic migration of shards from it to free more memory
redis_migrate_node_threshold_p	integer	Minimum free memory (excluding reserved memory) allowed on a node before automatic migration of shards from it to free more memory
redis_provision_node_threshold	integer	Minimum free memory (excluding reserved memory) allowed on a node before new shards can no longer be added to it
redis_provision_node_threshold_p	integer	Minimum free memory (excluding reserved memory) allowed on a node before new shards can no longer be added to it

Name	Type/Value	Description
redis_upgrade_policy	<code>major</code> <code>latest</code>	Create/upgrade Redis Enterprise software on databases in the cluster by compatibility with major versions or latest versions of OSS Redis
resp3_default	boolean (default: true)	Determines the default value of the <code>resp3</code> option upon upgrading a database to version 7.2
shards_overbooking	boolean	If true, all databases' <code>memory_size</code> is ignored during shards placement
show_internals	boolean	Show internal databases (and their shards and endpoints) REST APIs
slave_ha	boolean	Enable the replica high-availability mechanism
slave_ha_bdb_coldown_period	integer	Time in seconds between runs of the replica high-availability mechanism on different nodes on the same database
slave_ha_cooldown_period	integer	Time in seconds between runs of the replica high-availability mechanism on different nodes on the same database
slave_ha_grace_period	integer	Time in seconds between a node failure and when the replica high-availability mechanism starts relocating shards

Updated: August 17, 2023

CRDB object

An object that represents an Active-Active database.

Name	Type/Value	Description
guid	string	The global unique ID of the Active-Active database
causal_consistency	boolean	Enables causal consistency across CRDT instances
default_db_config	CRDB database_config object	Default database configuration
encryption	boolean	Encrypt communication
featureset_version	integer	Active-Active database active FeatureSet version
instances	array of CRDB instance_info objects	
local_databases	[{ bdb_uid : string, id : integer }, ...]	Mapping of instance IDs for local databases to local BDB IDs
name	string	Name of Active-Active database
protocol_version	integer	Active-Active database active protocol version

Updated: November 29, 2021

CRDB cluster info object

Configuration details for a cluster that is part of an Active-Active database.

Name	Type/Value	Description
------	------------	-------------

Name	Type/Value	Description
credentials	<pre>{ "username": string, "password": string }</pre>	Cluster access credentials (required)
name	string	Cluster fully qualified name, used to uniquely identify the cluster. Typically this is the same as the hostname used in the URL, although in some configurations the URL may point to a different name/address. (required)
replication_endpoint	string	Address to use for peer replication. If not specified, it is assumed that standard cluster naming conventions apply.
replication_tls_sni	string	Cluster SNI for TLS connections
url	string	Cluster access URL (required)

Updated: November 29, 2021

CRDB database config object

An object that represents the database configuration.

Name	Type/Value	Description
aof_policy	string	Policy for Append-Only File data persistence
authentication_admin_pass	string	Administrative databases access token
authentication_redis_pass	string	Redis AUTH password (deprecated as of Redis Enterprise v7.2, replaced with multiple passwords feature in version 6.0.X)
bigstore	boolean	Database driver is Auto Tiering
bigstore_ram_size	integer	Memory size of RAM size
data_persistence	string	Database on-disk persistence
enforce_client_authentication	'enabled' 'disabled'	Require authentication of client certificates for SSL connections to the database. If enabled, a certificate should be provided in either <code>authentication_ssl_client_certs</code> or <code>authentication_ssl_crft_certs</code>
max_aof_file_size	integer	Hint for maximum AOF file size
max_aof_load_time	integer	Hint for maximum AOF reload time
memory_size	integer	Database memory size limit, in bytes
oss_cluster	boolean	Enables OSS Cluster mode
oss_cluster_api_preferred_ip_type	string	Indicates preferred IP type in OSS cluster API: internal/external
oss_sharding	boolean	An alternative to shard_key_regex for using the common case of the OSS shard hashing policy
port	integer	TCP port for database access
proxy_policy	string	The policy used for proxy binding to the endpoint
rack_aware	boolean	Require the database to be always replicated across multiple racks
replication	boolean	Database replication

Name	Type/Value	Description
sharding	boolean (default: false)	Cluster mode (server-side sharding). When true, shard hashing rules must be provided by either oss_sharding or shard_key_regex
	[{ "regex": string }, ...]	
shard_key_regex	To use the default rules you should set the value to:	Custom keyname-based sharding rules (required if sharding is enabled)
	[{ "regex": ".*\\"{(?< tag >.*)\\"}.*"}, { "regex": "(?< tag >.)" }]	
shards_count	integer	Number of database shards
shards_placement	string	Control the density of shards: should they reside on as few or as many nodes as possible
snapshot_policy	array of snapshot_policy objects	Policy for snapshot-based data persistence (required)
tls_mode	string	Encrypt communication

Updated: August 17, 2023

CRDB health report object

An object that represents an Active-Active database health report.

Name	Type/Value	Description
active_config_version	integer	Active configuration version
cluster_name	string	Name of local Active-Active cluster
configurations	array of health_report_configuration objects	Stored database configurations
connection_error	string	Error string if remote cluster is not available

Name	Type/Value	Description
connections	<pre>[{ "name": string, "replication_links": [{ "link_uid": "bdb_uid:replica_uid", "status": "up down" }, "status": string }, ... }</pre>	Connections to other clusters and their statuses. A replication link's <code>bdb_uid</code> is the unique ID of a local database instance (<code>bdb</code>) in the current cluster. The <code>replica_uid</code> is the unique ID of the database's remote replica, located in the connected cluster.
name	string	Name of the Active-Active database

Updated: April 10, 2023

CRDB health report configuration object

An object that represents the database configuration to include in an Active-Active database health report.

Name	Type/Value	Description
causal_consistency	boolean	Enables causal consistency across Active-Active replicas
encryption	boolean	Intercluster encryption
featureset_version	integer	CRDB active FeatureSet version
instances	<pre>[{ "id": integer, // Unique instance ID "db_uid": string, // Local database instance ID "cluster": { "name": string // Cluster FQDN "url": string // Cluster access URL } }, ...]</pre>	Local database instances
name	string	Name of database
protocol_version	integer	CRDB active protocol version

Name	Type/Value	Description
status	string	<p>Current status of the configuration.</p> <p>Possible values:</p> <ul style="list-style-type: none"> posted: Configuration was posted to all replicas ready: All replicas have finished processing posted configuration (create a database) committed: Posted configuration is now active on all replicas commit-completed: All replicas have finished processing committed configuration (database is active) failed: Configuration failed to post
version	integer	Database configuration version

Updated: April 13, 2022

CRDB instance info object

An object that represents Active-Active instance info.

Name	Type/Value	Description
id	integer	Unique instance ID
cluster	CRDB cluster_info object	
compression	integer	Compression level when syncing from this source
db_config	CRDB database_config object	Database configuration
db_uid	string	ID of local database instance. This field is likely to be empty for instances other than the local one.

Updated: November 29, 2021

CRDB modify request object

An object to update an Active-Active database.

Name	Type/Value	Description
add_instances	array of CRDB instance_info objects	List of new CRDB instances
crdb	CRDB object	An object that represents an Active-Active database
force_update	boolean	(Warning: This flag can cause unintended and dangerous changes) Force the configuration update and increment the configuration version even if there is no change to the configuration parameters. If you use force, you can mistakenly cause the other instances to update to the configuration version even though it was not changed.
remove_instances	array of integers	List of unique instance IDs

Name	Type/Value	Description
remove_instances.force_remove	boolean	Force removal of instance from the Active-Active database. Before we remove an instance from an Active-Active database, all of the operations that the instance received from clients must be propagated to the other instances. This is the safe method to remove an instance from the Active-Active database. If the instance does not have connectivity to other instances, the propagation fails and removal fails. To remove an instance that does not have connectivity to other instances, you must use the force flag. The removed instance keeps its data and configuration for the instance. After you remove an instance by force, you must use the purge_instances API on the removed instance.

Updated: November 29, 2021

CRDB task object

An object that represents an Active-Active (CRDB) task.

Name	Type/Value	Description
id	string	CRDB task ID (read only)
crdb_guid	string	Globally unique Active-Active database ID (GUID) (read-only)
errors	[{ "cluster_name": string, "description": string, "error_code": string }, ...]	Details for errors that occurred on a cluster
status	'queued' 'started' 'finished' 'failed'	CRDB task status (read only)

Updated: November 29, 2021

Database alerts settings object

An API object that represents the database alerts configuration.

Name	Type/Value	Description
bdb_backup_delayed	bdb_alert_settings_with_threshold object	Periodic backup has been delayed for longer than specified threshold value (minutes)
bdb_crdt_src_high_syncer_lag	bdb_alert_settings_with_threshold object	CRDB source sync lag is higher than specified threshold value (seconds)
bdb_crdt_src_syncer_connection_error	bdb_alert_settings_with_threshold object	CRDB source sync had a connection error while trying to connect to replica source
bdb_crdt_src_syncer_general_error	bdb_alert_settings_with_threshold object	CRDB sync encountered in general error

Name	Type/Value	Description
bdb_high_latency	<code>bdb_alert_settings_with_threshold</code> object	Latency is higher than specified threshold value (microsec)
bdb_high_syncer_lag	<code>bdb_alert_settings_with_threshold</code> object	Replica of sync lag is higher than specified threshold value (seconds) (deprecated as of Redis Enterprise v5.0.1)
bdb_high_throughput	<code>bdb_alert_settings_with_threshold</code> object	Throughput is higher than specified threshold value (requests / sec)
bdb_long_running_action	<code>bdb_alert_settings_with_threshold</code> object	An alert for state machines that are running for too long
bdb_low_throughput	<code>bdb_alert_settings_with_threshold</code> object	Throughput is lower than specified threshold value (requests / sec)
bdb_ram_dataset_overhead	<code>bdb_alert_settings_with_threshold</code> object	Dataset RAM overhead of a shard has reached the threshold value (% of its RAM limit)
bdb_ram_values	<code>bdb_alert_settings_with_threshold</code> object	Percent of values kept in a shard's RAM is lower than (% of its key count)
bdb_replica_src_high_syncer_lag	<code>bdb_alert_settings_with_threshold</code> object	Replica of source sync lag is higher than specified threshold value (seconds)
bdb_replica_src_syncer_connection_error	<code>bdb_alert_settings_with_threshold</code> object	Replica of source sync has connection error while trying to connect replica source
bdb_replica_src_syncer_general_error	<code>bdb_alert_settings_with_threshold</code> object	Replica of sync encountered in general error
bdb_shard_num_ram_values	<code>bdb_alert_settings_with_threshold</code> object	Number of values kept in a shard's RAM is lower than (values)
bdb_size	<code>bdb_alert_settings_with_threshold</code> object	Dataset size has reached the threshold value (% of the memory limit)
bdb_syncer_connection_error	<code>bdb_alert_settings_with_threshold</code> object	Replica of sync has connection error while trying to connect replica source (deprecated as of Redis Enterprise v5.0.1)
bdb_syncer_general_error	<code>bdb_alert_settings_with_threshold</code> object	Replica of sync encountered in general error (deprecated as of Redis Enterprise v5.0.1)

Updated: August 17, 2023

BDB alert settings with threshold object

Name	Type/Value	Description
enabled	boolean (default: false)	Alert enabled or disabled
threshold	string	Threshold for alert going on/off

Updated: November 29, 2021

Database connection auditing configuration object

Database connection auditing configuration

Name	Type/Value	Description
audit_address	string	TCP/IP address where one can listen for notifications.
audit_port	integer	Port where one can listen for notifications.
audit_protocol	TCP local	Protocol used to process notifications. For production systems, TCP is the only valid value.

Name	Type/Value	Description
audit_reconnect_interval	integer	Interval (in seconds) between attempts to reconnect to the listener. Default is 1 second.
audit_reconnect_max_attempts	integer	Maximum number of attempts to reconnect. Default is 0 (infinite).

Updated: February 9, 2023

Job scheduler object

An API object that represents the job scheduler settings in the cluster.

Name	Type/Value	Description
backup_job_settings	backup_job_settings object	Backup job settings
cert_rotation_job_settings	cert_rotation_job_settings object	Job settings for internal certificate rotation
log_rotation_job_settings	log_rotation_job_settings object	Log rotation job settings
node_checks_job_settings	node_checks_job_settings object	Node checks job settings
redis_cleanup_job_settings	redis_cleanup_job_settings object	Redis cleanup job settings (deprecated as of Redis Enterprise v6.4.2, replaced with <code>persistence_cleanup_scan_interval</code>)
rotate_ccs_job_settings	rotate_ccs_job_settings object	Rotate CCS job settings

Updated: August 17, 2023

Backup job settings object

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the backup schedule

Updated: November 29, 2021

Certificate rotation job settings object

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the certificate rotation schedule
expiry_days_before_rotation	integer, (range: 1-90) (default: 60)	Number of days before a certificate expires before rotation

Updated: November 29, 2021

Log rotation job settings object

Name	Type/Value	Description
------	------------	-------------

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the log rotation schedule

Updated: November 29, 2021

Node checks job settings object

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the node checks schedule

Updated: November 29, 2021

Redis cleanup job settings object

Deprecated and replaced with `persistence_cleanup_scan_interval`.

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the Redis cleanup schedule

Updated: August 17, 2023

Rotate CCS job settings object

Name	Type/Value	Description
cron_expression	string	CRON expression that defines the CCS rotation schedule
file_suffix	string (default: 5min)	String added to the end of the rotated RDB files
rotate_max_num	integer, (range: 1-100) (default: 24)	The maximum number of saved RDB files

Updated: November 29, 2021

JWT authorize object

An API object for user authentication or a JW token refresh request.

Name	Type/Value	Description
password	string	The user's password (required)
ttl	integer (range: 1-86400) (default: 300)	Time to live - The amount of time in seconds the token will be valid
username	string	The user's username (required)

LDAP object

An API object that represents the cluster's [LDAP](#) configuration.

Name	Type/Value	Description
bind_dn	string	DN used when binding with the LDAP server to run queries
bind_pass	string	Password used when binding with the LDAP server to run queries
ca_cert	string	PEM-encoded CA certificate(s) used to validate TLS connections to the LDAP server
cache_ttl	integer (default: 300)	Maximum TTL (in seconds) of cached entries
control_plane	boolean (default: false)	Use LDAP for user authentication/authorization in the control plane
data_plane	boolean (default: false)	Use LDAP for user authentication/authorization in the data plane
dn_group_attr	string	The name of an attribute of the LDAP user entity that contains a list of the groups that user belongs to. (Mutually exclusive with "dn_group_query")
dn_group_query	complex object	An LDAP search query for mapping from a user DN to the groups the user is a member of. The substring "%D" in the filter will be replaced with the user's DN. (Mutually exclusive with "dn_group_attr")
starttls	boolean (default: false)	Use StartTLS negotiation for the LDAP connection
uris	array of strings	URIs of LDAP servers that only contain the schema, host, and port
user_dn_query	complex object	An LDAP search query for mapping from a username to a user DN. The substring "%u" in the filter will be replaced with the username. (Mutually exclusive with "user_dn_template")
user_dn_template	string	A string template that maps between the username, provided to the cluster for authentication, and the LDAP DN. The substring "%u" will be replaced with the username. (Mutually exclusive with "user_dn_query")

Updated: June 21, 2022

LDAP mapping object

An API object that represents an [LDAP mapping](#) between an LDAP group and [roles](#).

Name	Type/Value	Description
uid	integer	LDAP mapping's unique ID
account_id	integer	SM account ID
action_uid	string	Action UID. If it exists, progress can be tracked by the GET /actions/{uid} API (read-only)
b dbs_email_alerts	complex object	IDs of databases that associated email addresses will receive alerts for
cluster_email_alerts	boolean	Activate cluster email alerts for an associated email
dn	string	An LDAP group's distinguished name

Name	Type/Value	Description
email	string	Email address used for alerts (if set)
email_alerts	boolean (default: true)	Activate email alerts for an associated email
name	string	Role's name
role_uids	array of integers	List of role UIDs associated with the LDAP group

Updated: June 21, 2022

Module object

Represents a [Redis module](#).

Name	Type/Value	Description
uid	string	Cluster unique ID of module
architecture	string	Architecture used to compile the module
author	string	Module creator
capabilities	array of strings	List of capabilities supported by this module
command_line_args	string	Command line arguments passed to the module
config_command	string	Name of command to configure module arguments at runtime
dependencies	object dependencies	Module dependencies
description	string	Short description of the module
display_name	string	Name of module for display purposes
email	string	Author's email address
homepage	string	Module's homepage
is_bundled	boolean	Whether module came bundled with a version of Redis Enterprise
license	string	Module is distributed under this license
min_redis_pack_version	string	Minimum Redis pack version required by this module
min_redis_version	string	Minimum Redis version required by this module
module_file	string	Module filename
module_name	search ReJSON graph timeseries bf	Module's name
os	string	Operating system used to compile the module
os_list	array of strings	List of supported operating systems
semantic_version	string	Module's semantic version
sha256	string	SHA256 of module binary
version	integer	Module's version

Updated: August 17, 2023

Node object

An API object that represents a node in the cluster.

Name	Type/Value	Description
uid	integer	Cluster unique ID of node (read-only)
accept_servers	boolean (default: true)	The node only accepts new shards if accept_servers is true
addr	string	Internal IP address of node
architecture	string	Hardware architecture (read-only)
bigredis_storage_path	string	Flash storage path (read-only)
bigstore_driver	'ibm-capi-ga1' 'ibm-capi-ga2' 'ibm-capi-ga4' 'speedb' 'rocksdb'	Bigstore driver name or none (deprecated as of Redis Enterprise v7.2, use the cluster object 's bigstore_driver instead)
bigstore_size	integer	Storage size of bigstore storage (read-only)
cores	integer	Total number of CPU cores (read-only)
ephemeral_storage_path	string	Ephemeral storage path (read-only)
ephemeral_storage_size	number	Ephemeral storage size (bytes) (read-only)
external_addr	complex object	External IP addresses of node. GET / jsonschema to retrieve the object's structure.
max_listeners	integer	Maximum number of listeners on the node
max_redis_servers	integer	Maximum number of shards on the node
os_name	string	OS name (read-only)
os_semantic_version	string	Full version number (read-only)
os_version	string	Installed OS version (human-readable) (read-only)
persistent_storage_path	string	Persistent storage path (read-only)
persistent_storage_size	number	Persistent storage size (bytes) (read-only)
public_addr	string	Public IP address of node (deprecated as of Redis Enterprise v4.3.3, use external_addr instead)
rack_id	string	Rack ID where node is installed
recovery_path	string	Recovery files path
shard_count	integer	Number of shards on the node (read-only)
shard_list	array of integers	Cluster unique IDs of all node shards
software_version	string	Installed Redis Enterprise cluster software version (read-only)
status	'active' 'decommissioning' 'down' 'provisioning'	Node status (read-only)
supported_database_versions	[{ "db_type": string, "version": string }, ...]	Versions of open source databases supported by Redis Enterprise Software on the node (read-only) db_type: Type of database version: Version of database
system_time	string	System time (UTC) (read-only)
total_memory	integer	Total memory of node (bytes) (read-only)
uptime	integer	System uptime (seconds) (read-only)

Updated: August 24, 2023

OCSP object

An API object that represents the cluster's OCSP configuration.

Name	Type/Value	Description
ocsp_functionality	boolean (default: false)	Enables or turns off OCSP for the cluster
query_frequency	integer (range: 60-86400) (default: 3600)	The time interval in seconds between OCSP queries to check the certificate's status
recovery_frequency	integer (range: 60-86400) (default: 60)	The time interval in seconds between retries after the OCSP responder returns an invalid status for the certificate
recovery_max_tries	integer (range: 1-100) (default: 5)	The number of retries before the validation query fails and invalidates the certificate
responder_url	string	The OCSP server URL embedded in the proxy certificate (if available) (read-only)
response_timeout	integer (range: 1-60) (default: 1)	The time interval in seconds to wait for a response before timing out

Updated: August 10, 2022

OCSP status object

An API object that represents the cluster's OCSP status.

Name	Type/Value	Description
cert_status	string	Indicates the proxy certificate's status: GOOD/REVOKED/UNKNOWN (read-only)
responder_url	string	The OCSP responder URL this status came from (read-only)
next_update	string	The expected date and time of the next certificate status update (read-only)
produced_at	string	The date and time when the OCSP responder signed this response (read-only)
revocation_time	string	The date and time when the certificate was revoked or placed on hold (read-only)
this_update	string	The most recent time that the responder confirmed the current status (read-only)

Updated: August 10, 2022

Proxy object

An API object that represents a [proxy](#) in the cluster.

Name	Type/Value	Description
uid	integer	Unique ID of the proxy (read-only)
backlog	integer	TCP listen queue backlog
client_keepcnt	integer	Client TCP keepalive count
client_keepidle	integer	Client TCP keepalive idle
client_keepintvl	integer	Client TCP keepalive interval
conns	integer	Number of connections
duration_usage_threshold	integer, (range: 10-300)	Max number of threads
dynamic_threads_scaling	boolean	Automatically adjust the number of threads

Name	Type/Value	Description
ignore_bdb_cconn_limit	boolean	Ignore client connection limits
ignore_bdb_cconn_output_buff_limits	boolean	Ignore buffer limit
max_listeners	integer	Max number of listeners
max_servers	integer	Max number of Redis servers
max_threads	integer, (range: 1-256)	Max number of threads
max_worker_client_conns	integer	Max client connections per thread
max_worker_server_conns	integer	Max server connections per thread
max_worker_txns	integer	Max in-flight transactions per thread
threads	integer, (range: 1-256)	Number of threads
threads_usage_threshold	integer, (range: 50-99)	Max number of threads

Updated: November 29, 2021

Redis ACL object

An API object that represents a Redis [access control list \(ACL\)](#)

Name	Type/Value	Description
uid	integer	Object's unique ID
account_id	integer	SM account ID
acl	string	Redis ACL's string
action_uid	string	Action UID. If it exists, progress can be tracked by the GET /actions/{uid} API (read-only)
name	string	Redis ACL's name
min_version	string	Minimum database version that supports this ACL. Read only
max_version	string	Maximum database version that supports this ACL. Read only

Updated: August 17, 2023

Role object

An API object that represents a role.

Name	Type/Value	Description
uid	integer	Role's unique ID
account_id	integer	SM account ID
action_uid	string	Action UID. If it exists, progress can be tracked by the GET /actions/{uid} API (read-only)
management	'admin' 'db_member' 'db_viewer' 'cluster_member' 'cluster_viewer' 'none'	Management role
name	string	Role's name

Updated: November 29, 2021

Services configuration object

Optional cluster services settings

Name	Type/Value	Description
cm_server	cm_server object	Whether to enable/disable the CM server
crdb_coordinator	crdb_coordinator object	Whether to enable/disable the CRDB coordinator process
crdb_worker	crdb_worker object	Whether to enable/disable the CRDB worker processes
mdns_server	mdns_server object	Whether to enable/disable the multicast DNS server
pdns_server	pdns_server object	Whether to enable/disable the PDNS server
stats_archiver	stats_archiver object	Whether to enable/disable the stats archiver service

Updated: August 31, 2023

CM server object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the CM server

Updated: November 29, 2021

CRDB coordinator object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the CRDB coordinator process

Updated: November 29, 2021

CRDB worker object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the CRDB worker processes

Updated: November 29, 2021

MDNS server object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the multicast DNS server

Updated: November 29, 2021

PDNS server object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the PDNS server

Updated: November 29, 2021

Stats archiver object

Name	Type/Value	Description
operating_mode	'disabled' 'enabled'	Enable/disable the stats archiver service

Updated: November 29, 2021

Shard object

An API object that represents a Redis shard in a database.

Name	Type/Value	Description
uid	string	Cluster unique ID of shard
assigned_slots	string	Shards hash slot range
backup	backup object	Current status of scheduled periodic backup process
bdb_uid	integer	The ID of the database this shard belongs to
bigstore_ram_weight	number	Shards RAM distribution weight
detailed_status	'busy' 'down' 'importing' 'loading' 'ok' 'timeout' 'trimming' 'unknown'	A more detailed status of the shard
loading	loading object	Current status of dump file loading
node_uid	string	The ID of the node this shard is located on
redis_info	redis_info object	A sub-dictionary of the Redis INFO command

Name	Type/Value	Description
report_timestamp	string	The time in which the shard's info was collected (read-only)
role	'master' 'slave'	Role of this shard
status	'active' 'inactive' 'trimming'	The current status of the shard
sync	sync object	Shard's current sync status and progress

Updated: November 29, 2021

Backup object

Name	Type/Value	Description
progress	number, (range: 0-100)	Shard backup progress (percentage)
status	'exporting' 'succeeded' 'failed'	Status of scheduled periodic backup process

Updated: November 29, 2021

Loading object

Name	Type/Value	Description
progress	number, (range: 0-100)	Percentage of bytes already loaded
status	'in_progress' 'idle'	Status of the load of a dump file (read-only)

Updated: November 29, 2021

Sync object

Name	Type/Value	Description
progress	integer	Number of bytes remaining in current sync
status	'in_progress' 'idle' 'link_down'	Indication of the shard's current sync status

Updated: November 29, 2021

State machine object

A state machine object tracks the status of database actions.

A state machine contains the following attributes:

Name	Type/Value	Description
action_uid	string	A globally unique identifier of the action
object_name	string	Name of the object being manipulated by the state machine
status	pending	Requested state machine has not started
	active	State machine is currently running
	completed	Operation complete
	failed	Operation or state machine failed
name	string	Name of the running (or failed) state machine
state	string	Current state within the state machine, when known
error	string	A descriptive error string for failed state machine, when known

Updated: April 26, 2022

Statistics

Statistics overview

Clusters, databases, nodes, and shards collect various statistics at regular time intervals. You can view the statistics for these objects via GET `/stats` requests to their respective endpoints:

- [Cluster stats](#)
- [Database stats](#)
- [Node stats](#)
- [Shard stats](#)

Response object

Statistics returned from API requests always contain the following fields:

- **interval**: a string that represents the statistics time interval. Valid values include:
 - 1sec
 - 10sec
 - 5min
 - 15min
 - 1hour
 - 12hour
 - 1week
- **stime**: a timestamp that represents the beginning of the interval, in the format "2015-05-27T12:00:00Z"
- **etime**: a timestamp that represents the end of the interval, in the format "2015-05-27T12:00:00Z"

The statistics returned by the API also contain fields that represent the values of different metrics for an object during the specified time interval.

More details about the metrics relevant to each object:

- [Cluster metrics](#)
- [DB metrics](#)
- [Node metrics](#)
- [Shard metrics](#)



Note: Some statistics are for internal use only. They are not documented and should be ignored.



Note: Certain statistics will only appear in API responses when they are relevant.

Optional URL parameters

There are several optional URL parameters you can pass to the various GET stats requests in order to filter the returned statistics.

- `s time`: limit the start of the time range of the returned statistics
- `e time`: limit the end of the time range of the returned statistics
- `metrics`: only return the statistics for the specified metrics (comma-separated list)

Maximum number of samples per interval

The system retains a maximum number of most recent samples for each interval.

Interval	Max samples
1sec	10
10sec	30
5min	12
15min	96
1hour	168
12hour	62
1week	53

The actual number of samples returned by a GET stats request depends on how many samples are available and any filters applied by the optional URL parameters. For example, newly created objects (clusters, nodes, databases, or shards) or a narrow time filter range will return fewer samples.



Note: To reduce load generated by stats collection, relatively inactive databases or shards (less than 5 ops/sec) do not collect 1sec stats at one second intervals. Instead, they collect 1sec stats every 2-5 seconds but still retain the same maximum number of samples.

Updated: November 29, 2021

Cluster metrics

Metric name	Type	Description
available_flash	float	Sum of available flash in all nodes (bytes)
available_memory	float	Sum of available memory in all nodes (bytes)
avg_latency	float	Average latency of requests handled by all cluster endpoints (micro-sec); returned only when there is traffic
bigstore_free	float	Sum of free space of backend flash (used by flash DB's BigRedis) on all cluster nodes (bytes); only returned when BigRedis is enabled
bigstore_iops	float	Rate of I/O operations against backend flash for all shards which are part of a flash-based DB (BigRedis) in the cluster (ops/sec); returned only when BigRedis is enabled
bigstore_kv_ops	float	Rate of value read/write operations against backend flash for all shards which are part of a flash-based DB (BigRedis) in cluster (ops/sec); only returned when BigRedis is enabled
bigstore_throughput	float	Throughput I/O operations against backend flash for all shards which are part of a flash-based DB (BigRedis) in the cluster (bytes/sec); only returned when BigRedis is enabled
conns	float	Total number of clients connected to all cluster endpoints
cpu_idle	float	CPU idle time portion, the value is weighted between all nodes based on number of cores in each node (0-1, multiply by 100 to get percent)

Metric name	Type	Description
cpu_system	float	CPU time portion spent in kernel on the cluster, the value is weighted between all nodes based on number of cores in each node (0-1, multiply by 100 to get percent)
cpu_user	float	CPU time portion spent by users-space processes on the cluster. The value is weighted between all nodes based on number of cores in each node (0-1, multiply by 100 to get percent).
egress_bytes	float	Sum of rate of outgoing network traffic on all cluster nodes (bytes/sec)
ephemeral_storage_avail	float	Sum of disk space available to Redis Enterprise processes on configured ephemeral disk on all cluster nodes (bytes)
ephemeral_storage_free	float	Sum of free disk space on configured ephemeral disk on all cluster nodes (bytes)
free_memory	float	Sum of free memory in all cluster nodes (bytes)
ingress_bytes	float	Sum of rate of incoming network traffic on all cluster nodes (bytes/sec)
persistent_storage_avail	float	Sum of disk space available to Redis Enterprise processes on configured persistent disk on all cluster nodes (bytes)
persistent_storage_free	float	Sum of free disk space on configured persistent disk on all cluster nodes (bytes)
provisional_flash	float	Sum of provisional flash in all nodes (bytes)
provisional_memory	float	Sum of provisional memory in all nodes (bytes)
total_req	float	Request rate handled by all endpoints on the cluster (ops/sec)

Updated: November 29, 2021

DB metrics

Metric name	Type	Description
avg_latency	float	Average latency of operations on the DB (microseconds). Only returned when there is traffic.
avg_other_latency	float	Average latency of other (non read/write) operations (microseconds). Only returned when there is traffic.
avg_read_latency	float	Average latency of read operations (microseconds). Only returned when there is traffic.
avg_write_latency	float	Average latency of write operations (microseconds). Only returned when there is traffic.
big_del_flash	float	Rate of key deletes for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_del_ram	float	Rate of key deletes for keys in RAM (BigRedis) (key access/sec); this includes write misses (new keys created). Only returned when BigRedis is enabled.
big_fetch_flash	float	Rate of key reads/updates for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_fetch_ram	float	Rate of key reads/updates for keys in RAM (BigRedis) (key access/sec). Only returned when BigRedis is enabled.

Metric name	Type	Description
big_io_ratio_flash	float	Rate of key operations on flash. Can be used to compute the ratio of I/O operations (key access/sec). Only returned when BigRedis is enabled.
big_io_ratio_redis	float	Rate of Redis operations on keys. Can be used to compute the ratio of I/O operations (key access/sec). Only returned when BigRedis is enabled.
big_write_flash	float	Rate of key writes for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_write_ram	float	Rate of key writes for keys in RAM (BigRedis) (key access/sec); this includes write misses (new keys created). Only returned when BigRedis is enabled.
bigstore_io_dels	float	Rate of key deletions from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_io_read_bytes	float	Throughput of I/O read operations against backend flash
bigstore_io_reads	float	Rate of key reads from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_io_write_bytes	float	Throughput of I/O write operations against backend flash
bigstore_io_writes	float	Rate of key writes from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_iops	float	Rate of I/O operations against backend flash for all shards of the DB (BigRedis) (ops/sec). Only returned when BigRedis is enabled.
bigstore_kv_ops	float	Rate of value read/write/del operations against backend flash for all shards of the DB (BigRedis) (key access/sec). Only returned when BigRedis is enabled
bigstore_objs_flash	float	Value count on flash (BigRedis). Only returned when BigRedis is enabled.
bigstore_objs_ram	float	Value count in RAM (BigRedis). Only returned when BigRedis is enabled.
bigstore_throughput	float	Throughput of I/O operations against backend flash for all shards of the DB (BigRedis) (bytes/sec). Only returned when BigRedis is enabled.
conns	float	Number of client connections to the DB's endpoints
disk_frag_ratio	float	Flash fragmentation ratio (used/required). Only returned when BigRedis is enabled.
egress_bytes	float	Rate of outgoing network traffic to the DB's endpoint (bytes/sec)
evicted_objects	float	Rate of key evictions from DB (evictions/sec)
expired_objects	float	Rate keys expired in DB (expirations/sec)
fork_cpu_system	float	% cores utilization in system mode for all Redis shard fork child processes of this database
fork_cpu_user	float	% cores utilization in user mode for all Redis shard fork child processes of this database
ingress_bytes	float	Rate of incoming network traffic to the DB's endpoint (bytes/sec)
instantaneous_ops_per_sec	float	Request rate handled by all shards of the DB (ops/sec)
last_req_time	date, ISO_8601 format	Last request time received to the DB (ISO format 2015-07-05T22:16:18Z). Returns 1/1/1970 when unavailable.

Metric name	Type	Description
last_res_time	date, ISO_8601 format	Last response time received from DB (ISO format 2015-07-05T22:16:18Z). Returns 1/1/1970 when unavailable.
main_thread_cpu_system	float	% cores utilization in system mode for all Redis shard main threads of this database
main_thread_cpu_user	float	% cores utilization in user mode for all Redis shard main threads of this database
mem_frag_ratio	float	RAM fragmentation ratio (RSS/allocated RAM)
mem_not_counted_for_evict	float	Portion of used_memory (in bytes) not counted for eviction and OOM errors
mem_size_lua	float	Redis Lua scripting heap size (bytes)
monitor_sessions_count	float	Number of client connected in monitor mode to the DB
no_of_expires	float	Number of volatile keys in the DB
no_of_keys	float	Number of keys in the DB
other_req	float	Rate of other (non read/write) requests on DB (ops/sec)
other_res	float	Rate of other (non read/write) responses on DB (ops/sec)
pubsub_channels	float	Count the pub/sub channels with subscribed clients
pubsub_patterns	float	Count the pub/sub patterns with subscribed clients
ram_overhead	float	Non values RAM overhead (BigRedis) (bytes). Only returned when BigRedis is enabled.
read_hits	float	Rate of read operations accessing an existing key (ops/sec)
read_misses	float	Rate of read operations accessing a nonexistent key (ops/sec)
read_req	float	Rate of read requests on DB (ops/sec)
read_res	float	Rate of read responses on DB (ops/sec)
shard_cpu_system	float	% cores utilization in system mode for all Redis shard processes of this database
shard_cpu_user	float	% cores utilization in user mode for the Redis shard process
total_connections_received	float	Rate of new client connections to the DB (connections/sec)
total_req	float	Rate of all requests on DB (ops/sec)
total_res	float	Rate of all responses on DB (ops/sec)
used_bigstore	float	Flash used by DB (BigRedis) (bytes). Only returned when BigRedis is enabled.
used_memory	float	Memory used by DB (in BigRedis this includes flash) (bytes)
used_ram	float	RAM used by DB (BigRedis) (bytes). Only returned when BigRedis is enabled.
write_hits	float	Rate of write operations accessing an existing key (ops/sec)
write_misses	float	Rate of write operations accessing a nonexistent key (ops/sec)
write_req	float	Rate of write requests on DB (ops/sec)
write_res	float	Rate of write responses on DB (ops/sec)

Updated: November 29, 2021

Node metrics

Metric name	Type	Description
available_flash	float	Available flash on the node (bytes)
available_memory	float	Available RAM on the node (bytes)
avg_latency	float	Average latency of requests handled by endpoints on the node (micro-sec); returned only when there is traffic
bigstore_free	float	Free space of backend flash (used by flash DB's BigRedis) (bytes); returned only when BigRedis is enabled
bigstore_iops	float	Rate of I/O operations against backend flash for all shards which are part of a flash-based DB (BigRedis) on the node (ops/sec); returned only when BigRedis is enabled
bigstore_kv_ops	float	Rate of value read/write operations against backend flash for all shards which are part of a flash-based DB (BigRedis) on the node (ops/sec); returned only when BigRedis is enabled
bigstore_throughput	float	Throughput of I/O operations against backend flash for all shards which are part of a flash-based DB (BigRedis) on the node (bytes/sec); returned only when BigRedis is enabled
conns	float	Number of clients connected to endpoints on the node
cpu_idle	float	CPU idle time portion (0-1, multiply by 100 to get percent)
cpu_system	float	CPU time portion spent in kernel (0-1, multiply by 100 to get percent)
cpu_user	float	CPU time portion spent by users-space processes (0-1, multiply by 100 to get percent)
cur_aof_rewrites	float	Number of current AOF rewrites by shards on this node
egress_bytes	float	Rate of outgoing network traffic to the node (bytes/sec)
ephemeral_storage_avail	float	Disk space available to Redis Enterprise processes on configured ephemeral disk (bytes)
ephemeral_storage_free	float	Free disk space on configured ephemeral disk (bytes)
free_memory	float	Free memory on the node (bytes)
ingress_bytes	float	Rate of incoming network traffic to the node (bytes/sec)
persistent_storage_avail	float	Disk space available to Redis Enterprise processes on configured persistent disk (bytes)
persistent_storage_free	float	Free disk space on configured persistent disk (bytes)
provisional_flash	float	Amount of flash available for new shards on this node, taking into account overbooking, max Redis servers, reserved flash, and provision and migration thresholds (bytes)
provisional_memory	float	Amount of RAM available for new shards on this node, taking into account overbooking, max Redis servers, reserved memory, and provision and migration thresholds (bytes)
total_req	float	Request rate handled by endpoints on the node (ops/sec)

Shard metrics

Metric name	Type	Description
aof_rewrite_inprog	float	The number of simultaneous AOF rewrites that are in progress
avg_ttl	float	Estimated average time to live of a random key (msec)
big_del_flash	float	Rate of key deletes for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_del_ram	float	Rate of key deletes for keys in RAM (BigRedis) (key access/sec); this includes write misses (new keys created). Only returned when BigRedis is enabled.
big_fetch_flash	float	Rate of key reads/updates for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_fetch_ram	float	Rate of key reads/updates for keys in RAM (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_io_ratio_flash	float	Rate of key operations on flash. Can be used to compute the ratio of I/O operations (key access/sec). Only returned when BigRedis is enabled.
big_io_ratio_redis	float	Rate of Redis operations on keys. Can be used to compute the ratio of I/O operations (key access/sec). Only returned when BigRedis is enabled.
big_write_flash	float	Rate of key writes for keys on flash (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
big_write_ram	float	Rate of key writes for keys in RAM (BigRedis) (key access/sec); this includes write misses (new keys created). Only returned when BigRedis is enabled.
bigstore_io_dels	float	Rate of key deletions from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_io_read_bytes	float	Throughput of I/O read operations against backend flash for all shards of the DB (BigRedis) (bytes/sec). Only returned when BigRedis is enabled.
bigstore_io_reads	float	Rate of key reads from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_io_write_bytes	float	Throughput of I/O write operations against backend flash for all shards of the DB (BigRedis) (bytes/sec). Only returned when BigRedis is enabled.
bigstore_io_writes	float	Rate of key writes from flash (key access/sec). Only returned when BigRedis is enabled.
bigstore_iops	float	Rate of I/O operations against backend flash for all shards of the DB (BigRedis) (ops/sec). Only returned when BigRedis is enabled.
bigstore_kv_ops	float	Rate of value read/write/del operations against backend flash for all shards of the DB (BigRedis) (key access/sec). Only returned when BigRedis is enabled.
bigstore_objs_flash	float	Key count on flash (BigRedis). Only returned when BigRedis is enabled.
bigstore_objs_ram	float	Key count in RAM (BigRedis). Only returned when BigRedis is enabled.

Metric name	Type	Description
bigstore_throughput	float	Throughput of I/O operations against backend flash for all shards of the DB (BigRedis) (bytes/sec). Only returned when BigRedis is enabled.
blocked_clients	float	Count the clients waiting on a blocking call
connected_clients	float	Number of client connections to the specific shard
disk_frag_ratio	float	Flash fragmentation ratio (used/required). Only returned when BigRedis is enabled.
evicted_objects	float	Rate of key evictions from DB (evictions/sec)
expired_objects	float	Rate keys expired in DB (expirations/sec)
fork_cpu_system	float	% cores utilization in system mode for the Redis shard fork child process
fork_cpu_user	float	% cores utilization in user mode for the Redis shard fork child process
last_save_time	float	Time of the last RDB save
main_thread_cpu_system	float	% cores utilization in system mode for the Redis shard main thread
main_thread_cpu_user	float	% cores utilization in user mode for the Redis shard main thread
mem_frag_ratio	float	RAM fragmentation ratio (RSS/allocated RAM)
mem_not_counted_for_evict	float	Portion of used_memory (in bytes) not counted for eviction and OOM errors
mem_size_lua	float	Redis Lua scripting heap size (bytes)
no_of_expires	float	Number of volatile keys on the shard
no_of_keys	float	Number of keys in DB
pubsub_channels	float	Count the pub/sub channels with subscribed clients
pubsub_patterns	float	Count the pub/sub patterns with subscribed clients
rdb_changes_since_last_save	float	Count changes since last RDB save
read_hits	float	Rate of read operations accessing an existing key (ops/sec)
read_misses	float	Rate of read operations accessing a nonexistent key (ops/sec)
shard_cpu_system	float	% cores utilization in system mode for the Redis shard process
shard_cpu_user	float	% cores utilization in user mode for the Redis shard process
total_req	float	Rate of operations on DB (ops/sec)
used_memory	float	Memory used by shard (in BigRedis this includes flash) (bytes)
used_memory_peak	float	The largest amount of memory used by this shard (bytes)
used_memory_rss	float	Resident set size of this shard (bytes)
write_hits	float	Rate of write operations accessing an existing key (ops/sec)
write_misses	float	Rate of write operations accessing a nonexistent key (ops/sec)

Updated: November 29, 2021

Suffix object

An API object that represents a DNS suffix in the cluster.

Name	Type/Value	Description
default	boolean	Suffix is the default suffix for the cluster (read-only)
internal	boolean	Does the suffix point to internal IP addresses (read-only)
mdns	boolean	Support for multicast DNS (read-only)
name	string	Unique suffix name that represents its zone (read-only)
slaves	array of strings	Frontend DNS servers to be updated by this suffix
use_aaaa_ns	boolean	Suffix uses AAAA NS entries (read-only)

Updated: November 29, 2021

User object

An API object that represents a Redis Enterprise user.

Name	Type/Value	Description
uid	integer	User's unique ID
account_id	integer	SM account ID
action_uid	string	Action UID. If it exists, progress can be tracked by the GET /actions/{uid} API request (read-only)
auth_method	'regular'	User's authentication method (deprecated as of Redis Enterprise v7.2)
b dbs_email_alerts	complex object	UIDs of databases that user will receive alerts for
cluster_email_alerts	boolean	Activate cluster email alerts for a user
email	string	User's email (pattern matching only ASCII characters)
email_alerts	boolean (default: true)	Activate email alerts for a user
name	string	User's name (pattern does not allow non-ASCII and special characters &,<,>,")
password	string	User's password. Note that it could also be an already hashed value, in which case the password_hash_method parameter is also provided.
password_hash_method	'1'	Used when password is passed pre-hashed to specify the hashing method
password_issue_date	string	The date in which the password was set (read-only)
role	'admin' 'cluster_member' 'cluster_viewer' 'db_member' 'db_viewer' 'none'	User's role
role_uids	array of integers	List of role UIDs associated with the LDAP group

Updated: August 17, 2023

Permissions

Some Redis Enterprise REST API requests may require the user to have specific permissions.

Administrators can assign a predefined role to a user via the [admin console](#) or a [PUT /users/{uid}](#) API request in order to grant necessary permissions

to them.

Roles

Each user in the cluster has an assigned cluster management role, which defines the permissions granted to the user.

Available management roles include:

- **none**: No REST API permissions.
- **db_viewer**: Can view database info.
- **db_member**: Can create or modify databases and view their info.
- **cluster_viewer**: Can view cluster and database info.
- **cluster_member**: Can modify the cluster and databases and view their info.
- **admin**: Can view and modify all elements of the cluster.

Permissions list for each role

Role	Permissions
none	No permissions
admin	<code>add_cluster_module, cancel_cluster_action, cancel_node_action, config_ldap, config_ocsp, create_bdb, create_crdb, create_ldap_mapping, create_new_user, create_redis_acl, create_role, delete_bdb, delete_cluster_module, delete_crdb, delete_ldap_mapping, delete_redis_acl, delete_role, delete_user, edit_bdb_module, flush_crdb, install_new_license, migrate_shard, purge_instance, reset_bdb_current_backup_status, reset_bdb_current_export_status, reset_bdb_current_import_status, start_bdb_export, start_bdb_import, start_cluster_action, start_node_action, test_ocsp_status, update_bdb, update_bdb_alerts, update_bdb_with_action, update_cluster, update_crdb, update_ldap_mapping, update_node, update_proxy, update_redis_acl, update_role, update_user, view_all_bdb_stats, view_all_bdb_alerts, view_all_bdb_info, view_all_ldap_mappings_info, view_all_nodes_alerts, view_all_nodes_checks, view_all_nodes_info, view_all_nodes_stats, view_all_proxies_info, view_all_redis_acls_info, view_all_roles_info, view_all_shard_stats, view_all_users_info, view_bdb_alerts, view_bdb_info, view_bdb_stats, view_cluster_alerts, view_cluster_info, view_cluster_keys, view_cluster_modules, view_cluster_stats, view_crdb, view_crdb_list, view_endpoint_stats, view_ldap_config, view_ldap_mapping_info, view_license, view_logged_events, view_node_alerts, view_node_check, view_node_info, view_node_stats, view_ocsp_config, view_ocsp_status, view_proxy_info, view_redis_acl_info, view_redis_pass, view_role_info, view_shard_stats, view_status_of_all_node_actions, view_status_of_cluster_action, view_status_of_node_action, view_user_info</code>
cluster_member	<code>create_bdb, create_crdb, delete_bdb, delete_crdb, edit_bdb_module, flush_crdb, migrate_shard, purge_instance, reset_bdb_current_backup_status, reset_bdb_current_export_status, reset_bdb_current_import_status, start_bdb_export, start_bdb_import, update_bdb, update_bdb_alerts, update_bdb_with_action, update_crdb, view_all_bdb_stats, view_all_bdb_alerts, view_all_bdb_info, view_all_nodes_alerts, view_all_nodes_checks, view_all_nodes_info, view_all_nodes_stats, view_all_proxies_info, view_all_redis_acls_info, view_all_roles_info, view_all_shard_stats, view_bdb_alerts, view_bdb_info, view_bdb_stats, view_cluster_alerts, view_cluster_info, view_cluster_keys, view_cluster_modules, view_cluster_stats, view_crdb, view_crdb_list, view_endpoint_stats, view_license, view_logged_events, view_node_alerts, view_node_check, view_node_info, view_node_stats, view_proxy_info, view_redis_acl_info, view_redis_pass, view_role_info, view_shard_stats, view_status_of_all_node_actions, view_status_of_cluster_action, view_status_of_node_action</code>

Role	Permissions
cluster_viewer	view_all_bdb_stats, view_all_bdbs_alerts, view_all_bdbs_info, view_all_nodes_alerts, view_all_nodes_checks, view_all_nodes_info, view_all_nodes_stats, view_all_proxies_info, view_all_redis_acls_info, view_all_roles_info, view_all_shard_stats, view_bdb_alerts, view_bdb_info, view_bdb_stats, view_cluster_alerts, view_cluster_info, view_cluster_modules, view_cluster_stats, view_crdb, view_crdb_list, view_endpoint_stats, view_license, view_logged_events, view_node_alerts, view_node_check, view_node_info, view_node_stats, view_proxy_info, view_redis_acl_info, view_role_info, view_shard_stats, view_status_of_all_node_actions, view_status_of_cluster_action, view_status_of_node_action
db_member	create_bdb, create_crdb, delete_bdb, delete_crdb, edit_bdb_module, flush_crdb, migrate_shard, purge_instance, reset_bdb_current_backup_status, reset_bdb_current_export_status, reset_bdb_current_import_status, start_bdb_export, start_bdb_import, update_bdb, update_bdb_alerts, update_bdb_with_action, update_crdb, view_all_bdb_stats, view_all_bdbs_alerts, view_all_bdbs_info, view_all_nodes_alerts, view_all_nodes_checks, view_all_nodes_info, view_all_nodes_stats, view_all_proxies_info, view_all_redis_acls_info, view_all_roles_info, view_all_shard_stats, view_bdb_alerts, view_bdb_info, view_bdb_stats, view_cluster_alerts, view_cluster_info, view_cluster_modules, view_cluster_stats, view_crdb, view_crdb_list, view_endpoint_stats, view_license, view_logged_events, view_node_alerts, view_node_check, view_node_info, view_node_stats, view_proxy_info, view_redis_acl_info, view_redis_pass, view_role_info, view_shard_stats, view_status_of_all_node_actions, view_status_of_cluster_action, view_status_of_node_action
db_viewer	view_all_bdb_stats, view_all_bdbs_alerts, view_all_bdbs_info, view_all_nodes_alerts, view_all_nodes_checks, view_all_nodes_info, view_all_nodes_stats, view_all_proxies_info, view_all_redis_acls_info, view_all_roles_info, view_all_shard_stats, view_bdb_alerts, view_bdb_info, view_bdb_stats, view_cluster_alerts, view_cluster_info, view_cluster_modules, view_cluster_stats, view_crdb, view_crdb_list, view_endpoint_stats, view_license, view_node_alerts, view_node_check, view_node_info, view_node_stats, view_proxy_info, view_redis_acl_info, view_role_info, view_shard_stats, view_status_of_all_node_actions, view_status_of_cluster_action, view_status_of_node_action

Roles list per permission

Permission	Roles
add_cluster_module	admin
cancel_cluster_action	admin
cancel_node_action	admin
config_ldap	admin
config_ocsp	admin
create_bdb	admin cluster_member db_member
create_crdb	admin cluster_member db_member
create_ldap_mapping	admin
create_new_user	admin
create_redis_acl	admin
create_role	admin
delete_bdb	admin cluster_member db_member
delete_cluster_module	admin

Permission	Roles
delete_crdb	admin cluster_member db_member
delete_ldap_mapping	admin
delete_redis_acl	admin
delete_role	admin
delete_user	admin
edit_bdb_module	admin cluster_member db_member
flush_crdb	admin cluster_member db_member
install_new_license	admin
migrate_shard	admin cluster_member db_member
purge_instance	admin cluster_member db_member
reset_bdb_current_backup_status	admin cluster_member db_member
reset_bdb_current_export_status	admin cluster_member db_member
reset_bdb_current_import_status	admin cluster_member db_member
start_bdb_export	admin cluster_member db_member
start_bdb_import	admin cluster_member db_member
start_cluster_action	admin
start_node_action	admin
test_ocsp_status	admin
update_bdb	admin cluster_member db_member
update_bdb_alerts	admin cluster_member db_member
update_bdb_with_action	admin cluster_member db_member
update_cluster	admin
update_crdb	admin cluster_member db_member
update_ldap_mapping	admin
update_node	admin
update_proxy	admin
update_redis_acl	admin
update_role	admin
update_user	admin

Permission	Roles
view_all_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer
view_all_bdb_alerts	admin cluster_member cluster_viewer db_member db_viewer
view_all_bdb_info	admin cluster_member cluster_viewer db_member db_viewer
view_all_ldap_mappings_info	admin
view_all_nodes_alerts	admin cluster_member cluster_viewer db_member db_viewer
view_all_nodes_checks	admin cluster_member cluster_viewer db_member db_viewer
view_all_nodes_info	admin cluster_member cluster_viewer db_member db_viewer
view_all_nodes_stats	admin cluster_member cluster_viewer db_member db_viewer
view_all_proxies_info	admin cluster_member cluster_viewer db_member db_viewer
view_all_redis_acls_info	admin cluster_member cluster_viewer db_member db_viewer
view_all_roles_info	admin cluster_member cluster_viewer db_member db_viewer
view_all_shard_stats	admin cluster_member cluster_viewer db_member db_viewer
view_all_users_info	admin
view_bdb_alerts	admin cluster_member cluster_viewer db_member db_viewer

Permission	Roles
view_bdb_info	admin cluster_member cluster_viewer db_member db_viewer
view_bdb_stats	admin cluster_member cluster_viewer db_member db_viewer
view_cluster_alerts	admin cluster_member cluster_viewer db_member db_viewer
view_cluster_info	admin cluster_member cluster_viewer db_member db_viewer
view_cluster_keys	admin cluster_member
view_cluster_modules	admin cluster_member cluster_viewer db_member db_viewer
view_cluster_stats	admin cluster_member cluster_viewer db_member db_viewer
view_crdb	admin cluster_member cluster_viewer db_member db_viewer
view_crdb_list	admin cluster_member cluster_viewer db_member db_viewer
view_endpoint_stats	admin cluster_member cluster_viewer db_member db_viewer
view_ldap_config	admin
view_ldap_mapping_info	admin
view_license	admin cluster_member cluster_viewer db_member db_viewer
view_logged_events	admin cluster_member cluster_viewer db_member
view_node_alerts	admin cluster_member cluster_viewer db_member db_viewer

Permission	Roles
view_node_check	admin cluster_member cluster_viewer db_member db_viewer
view_node_info	admin cluster_member cluster_viewer db_member db_viewer
view_node_stats	admin cluster_member cluster_viewer db_member db_viewer
view_ocsp_config	admin
view_ocsp_status	admin
view_proxy_info	admin cluster_member cluster_viewer db_member db_viewer
view_redis_acl_info	admin cluster_member cluster_viewer db_member db_viewer
view_redis_pass	admin cluster_member db_member
view_role_info	admin cluster_member cluster_viewer db_member db_viewer
view_shard_stats	admin cluster_member cluster_viewer db_member db_viewer
view_status_of_all_node_actions	admin cluster_member cluster_viewer db_member db_viewer
view_status_of_cluster_action	admin cluster_member cluster_viewer db_member db_viewer
view_status_of_node_action	admin cluster_member cluster_viewer db_member db_viewer
view_user_info	admin

Updated: August 17, 2023

Terminology in Redis Enterprise Software

Here are explanations of some of the terms used in Redis Enterprise Software.

Node

A **node** is a physical machine, virtual machine, container or cloud instance on which the RS installation package was installed and the setup process was run in order to make the machine part of the cluster.

Each node is a container for running multiple open source Redis instances, referred to as “shards”.

The recommended configuration for a production cluster is an uneven number of nodes, with a minimum of three. Note that in some configurations, certain functionalities might be blocked. For example, if a cluster has only one node you cannot enable database replication, which helps to achieve high availability.

A node is made up of several components, as detailed below, and works together with the other cluster nodes.

Redis instance (shard)

As indicated above, each node serves as a container for hosting multiple database instances, referred to as “shards”.

Redis Enterprise Software supports various database configurations:

- **Standard Redis database** - A single Redis shard with no replication or clustering.
- **Highly available Redis database** - Every database master shard has a replica shard, so that if the master shard fails the cluster can automatically fail over to the replica with minimal impact. Master and replica shards are always placed on separate nodes to ensure high availability.
- **Clustered Redis database** - The data stored in the database is split across several shards. The number of shards can be defined by the user. Various performance optimization algorithms define where shards are placed within the cluster. During the lifetime of the cluster, these algorithms might migrate a shard between nodes.
- **Clustered and highly available Redis database** - Each master shard in the clustered database has a replica shard, enabling failover if the master shard fails.

Proxy

Each node includes one zero-latency, multi-threaded proxy (written in low-level C) that masks the underlying system complexity. The proxy oversees forwarding Redis operations to the database shards on behalf of a Redis client.

The proxy simplifies the cluster operation, from the application or Redis client point of view, by enabling the use of a standard Redis client. The zero-latency proxy is built over a cut-through architecture and employs various optimization methods. For example, to help ensure high-throughput and low-latency performance, the proxy might use instruction pipelining even if not instructed to do so by the client.

Database endpoint

Each database is served by a database endpoint that is part of and managed by the proxies. The endpoint oversees forwarding Redis operations to specific database shards.

If the master shard fails and the replica shard is promoted to master, the master endpoint is updated to point to the new master shard.

If the master endpoint fails, the replica endpoint is promoted to be the new master endpoint and is updated to point to the master shard.

Similarly, if both the master shard and the master endpoint fail, then both the replica shard and the replica endpoint are promoted to be the new master shard and master endpoint.

Shards and their endpoints do not have to reside within the same node in the cluster.

In the case of a clustered database with multiple database shards, only one master endpoint acts as the master endpoint for all master shards, forwarding Redis operations to all shards as needed.

Cluster manager

The cluster manager oversees all node management-related tasks, and the cluster manager in the master node looks after all the cluster related tasks.

The cluster manager is designed in a way that is totally decoupled from the Redis operation. This enables RS to react in a much faster and accurate manner to failure events, so that, for example, a node failure event triggers mass failover operations of all the master endpoints and master shards that are hosted on the failed node.

In addition, this architecture guarantees that each Redis shard is only dealing with processing Redis commands in a shared-nothing architecture, thus maintaining the inherent high-throughput and low-latency of each Redis process. Lastly, this architecture guarantees that any change in the cluster manager itself does not affect the Redis operation.

Some of the primary functionalities of the cluster manager include:

- Deciding where shards are created
- Deciding when shards are migrated and to where
- Monitoring database size
- Monitoring databases and endpoints across all nodes
- Running the database resharding process
- Running the database provisioning and de-provisioning processes
- Gathering operational statistics
- Enforcing license and subscription limitations

Updated: August 31, 2022

Release notes

Here's what changed recently in Redis Enterprise Software:

Version (Release date)	Major changes	OSS Redis compatibility
7.2.4 releases	Redis 7.0 and 7.2 features. Auto Tiering (enhanced successor to Redis on Flash). RESP3 support. Sharded pub/sub. Preview of the new Cluster Manager UI. Redis Stack 7.2 features. Three Redis database versions. License file structure updates. Redis ACL selectors and enhanced key-based permissions. New INFO fields. Log rotation enhancements. Multi-OS upgrade support for clusters with modules.	Redis 7.2.0
6.4.2 releases	Pub/sub ACLs & default permissions. Validate client certificates by subject attributes. Ubuntu 20.04 support.	Redis 6.2.10
6.2.18 releases	Database auditing. Private key encryption. Active-Active database support for MEMORY USAGE command.	Redis 6.2.6
6.2.12 (August 2022)	OCSP Support. Password & session configuration changes. RHEL 8.6 support.	Redis 6.2.6
6.2.10 (February 2022)	Python 3 support. RHEL 8.5 support.	Redis 6.2.5
6.2.8 (October 2021)	RHEL 8 support. Set backup start time.	Redis 6.2.3
6.2.4 (August 2021)	Internode encryption. Nginx replaced by envoy. New upgrade policies/behavior.	Redis 6.2
6.0.20 (April 2021)	Role-based LDAP integration. Enhanced client mutual authentication. Active-Active improvements for eviction policies, migration, and the BITFIELD data type.	Redis 6.0.9
6.0.12 (January 2021)	Distribute synchronization across nodes for Active-Active and Active-Passive databases. Disable internal services to free memory. User accounts support password rotation. Module dependencies automatically installed. Syncer process recovery.	Redis 6.0.6
6.0.8 (September 2020)	RediSearch 2.0 support. Improved rladmin support for module upgrades.	Redis 6.0.5
6.0 (May 2020)	ACL and RBAC improvements for database access. Active-Active databases support Redis Streams.	Redis 6
Previous releases	Release notes for Redis Enterprise Software 5.6.0 (April 2020) and earlier versions.	

Updated: November 12, 2021

Redis Enterprise Software release notes 7.2.4

Redis Enterprise Software version 7.2.4 is now available!

Highlights

This version offers:

- Redis 7.0 and 7.2 features
- Auto Tiering (enhanced successor to Redis on Flash)
- RESP3 support
- Sharded pub/sub
- A preview of the new Cluster Manager UI (admin console)
- Redis Stack 7.2 features
- Three Redis database versions: 7.2, 6.2, 6.0
- License file structure updates
- Redis ACL selectors and enhanced key-based permissions
- New INFO fields
- Log rotation enhancements
- Multi-OS upgrade support for clusters with modules

Detailed release notes

For more detailed release notes, select a build version from the following table:

Version (Release date)	Major changes	OSS Redis compatibility
7.2.4-52 (August 2023)	Redis 7.0 and 7.2 features. Auto Tiering (enhanced successor to Redis on Flash). RESP3 support. Sharded pub/sub. Preview of the new Cluster Manager UI. Redis Stack 7.2 features. Three Redis database versions. License file structure updates. Redis ACL selectors and enhanced key-based permissions. New INFO fields. Log rotation enhancements. Multi-OS upgrade support for clusters with modules.	Redis 7.2.0

Version changes

Breaking changes

For a list of potentially breaking changes introduced in version 7.2, see:

- [Breaking changes](#)
- [Redis 7.2 breaking changes](#)

To prevent potential application issues due to RESP3 breaking changes, see [Client prerequisites for Redis 7.2 upgrade](#).

Deprecations

Command deprecations

- [CLUSTER SLOTS](#) is deprecated as of Redis 7.0
- [JSON .RESP](#) is deprecated as of Redis Stack 7.2.

- `QUIT` is deprecated as of Redis 7.2

API deprecations

Fields deprecated as of Redis Enterprise v4.3.3:

- `smtp_use_tls` (replaced with `smtp_tls_mode`)
- `dns_address_master`
- `endpoint_node`
- `endpoint_ip`
- `public_addr` (replaced with `external_addr`)

Fields deprecated as of Redis Enterprise v4.4.2:

- `default_shards_overbooking` (replaced with `shards_overbooking`)

Fields deprecated as of Redis Enterprise v6.4.2:

- `use_ipv6` (replaced with `use_external_ipv6`)
- `redis_cleanup_job_settings` (replaced with `persistence_cleanup_scan_interval`)

Fields deprecated as of Redis Enterprise v5.0.1:

- `bdb_high_syncer_lag` (replaced with `replica_src_high_syncer_lag` and `crdt_src_high_syncer_lag`)
- `bdb_syncer_connection_error`
- `bdb_syncer_general_error`
- `sync_sources` (replaced with `replica_sources` and `crdt_sources`)
- `sync` (replaced with `replica_sync` and `crdt_sync`)
- `ssl` (replaced with `tls_mode`)

Fields deprecated as of Redis Enterprise v7.2.4:

- `node.bigstore_driver` (replaced with `cluster.bigstore_driver`)
- `auth_method`
- `authentication_redis_pass` (replaced with multiple passwords feature in version 6.0.X)

Other deprecated fields:

- `import/rdb_url` (deprecated as of Redis Enterprise v4.X)
- `logrotate_dir` (to be replaced with `logrotate_config` or removed)

Deprecated CLI commands:

- `rlutil change_master` (deprecated as of Redis Enterprise v6.2.18, replaced with `rladmin change_master`)
- `rlutil reserved_ports` (deprecated as of Redis Enterprise v7.2.4, replaced with `rladmin cluster config reserved_ports`)

REST API requests deprecated as of Redis Enterprise v7.2.4:

- `POST /v1/modules` (replaced with `POST /v2/modules`)
- `DELETE /v1/modules` (replaced with `DELETE /v2/modules`)

Access control deprecations

- The following predefined roles and Redis ACLs are not available after upgrading to Redis Enterprise Software version 7.2.4 if they are not associated with any users or databases in the cluster:
 - Custom roles (not management roles): Cluster Member, Cluster Viewer, DB Member, DB Viewer, None.

- Redis ACLs: Not Dangerous and Read Only.
- A deprecation notice for SASL-based LDAP was included in [previous Redis Enterprise Software release notes](#). When you upgrade to Redis Enterprise Software version 7.2.4, all existing “external” users (previously used to support SASL-based LDAP) will be removed.

Legacy UI

With the release of the new Cluster Manager UI, the legacy UI is considered deprecated and will eventually be phased out. New functionality will only be implemented in the new Cluster Manager UI, and the old UI will no longer be maintained except for critical bug fixes.

RedisGraph

Redis has announced the end of life for RedisGraph. Redis will continue to support all RedisGraph customers, including releasing patch versions until January 31, 2025.

See the [RedisGraph end-of-life announcement](#) for more details.

RHEL and CentOS 7.0-7.9

Support for RHEL and CentOS 7.0-7.9 is considered deprecated and will be removed in a future release.

Oracle Linux 7

Oracle Linux 7 support is considered deprecated and will be removed in a future release.

Amazon Linux 1

Amazon Linux 1 support is considered deprecated and will be removed in a future release.

Ubuntu 16.04

The deprecation of Ubuntu 16.04 was announced in the [Redis Enterprise Software 6.4.2 release notes](#). As of Redis Enterprise Software 7.2.4, Ubuntu 16.04 is no longer supported.

RC4 encryption cipher

The RC4 encryption cipher is considered deprecated in favor of stronger ciphers. Support for RC4 by the [discovery service](#) will be removed in a future release.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES. Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later. Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

Supported platforms

 Supported – The platform is supported for this version of Redis Enterprise Software.

 Deprecated – The platform is still supported for this version of Redis Enterprise Software, but support will be removed in a future release.

 End of life – Platform support ended in this version of Redis Enterprise Software.

Redis Enterprise	7.2.4	6.4.2	6.2.18	6.2.12	6.2.10	6.2.8	6.2.4
Ubuntu¹							
20.04			-	-	-	-	-
18.04							
16.04							

Redis Enterprise	7.2.4	6.4.2	6.2.18	6.2.12	6.2.10	6.2.8	6.2.4
RHEL & CentOS²							
8.8	?	-	-	-	-	-	-
8.7	?	?	-	-	-	-	-
8.5-8.6	?	?	?	?	?	-	-
8.0-8.4	?	?	?	?	?	?	-
7.0-7.9	△	?	?	?	?	?	?
Oracle Linux³							
8	?	?	?	?	?	-	-
7	△	?	?	?	?	?	?
Rocky Linux³							
8	?	?	?	-	-	-	-
Amazon Linux							
2	?	?	-	-	-	-	-
1	△	?	?	?	?	?	?
Docker⁴							
Kubernetes⁵							

1. The server version of Ubuntu is recommended for production installations. The desktop version is only recommended for development deployments.

2. RHEL and CentOS deployments require OpenSSL 1.0.2 and [firewall configuration](#).

3. Based on the corresponding RHEL version.

4. [Docker images](#) of Redis Enterprise Software are certified for development and testing only.

5. See the [Redis Enterprise for Kubernetes documentation](#).

6. Ubuntu 20.04 support was added in Redis Enterprise Software [6.4.2-43](#).

7. A release candidate for Amazon Linux 2 support was added in Redis Enterprise Software [6.4.2-61](#). Official support for Amazon Linux 2 was added in Redis Enterprise Software [6.4.2-69](#).

Known limitations

Command limitations

- [CLIENT NO-TOUCH](#) might not run correctly in the following cases:

- The Redis database version is earlier than 7.2.0.
- The [CLIENT NO-TOUCH](#) command is forbidden by ACL rules.

Before sending this command, clients should verify the database version is 7.2.0 or later and that using this command is allowed.

- You cannot use [SUNSUBSCRIBE](#) to unsubscribe from a shard channel if the regex changed while subscribed.
- Using [XREADGROUP BLOCK](#) with `>` to return all new streams will cause the Redis database to freeze until the shard is restarted. ([#12031](#))
- Because a rejected command does not record the duration for command stats, an error will appear after it is reprocessed that will cause the Redis database to freeze until the shard is restarted. ([#12247](#))

Modules cannot load in Oracle Linux 7 & 8

Databases hosted on Oracle Linux 7 & 8 cannot load modules.

As a temporary workaround, you can change the node's `os_name` in the Cluster Configuration Store (CCS):

```
ccs-cli hset node:<ID> os_name rhel
```

Cluster recovery with manually uploaded modules

For clusters containing databases with manually uploaded modules, [cluster recovery](#) requires an extra step.

After installing Redis Enterprise Software on the cluster nodes, upload compatible modules to `modulesdir` (`/opt/redislabs/lib/modules`) before continuing the recovery process.

This limitation will be removed in a future maintenance release.

Updated: August 31, 2023

Redis Enterprise Software release notes 7.2.4-52 (August 2023)

Redis Enterprise Software version 7.2.4 is now available!

Highlights

This version offers:

- Redis 7.0 and 7.2 features
- Auto Tiering (enhanced successor to Redis on Flash)
- RESP3 support
- Sharded pub/sub
- A preview of the new Cluster Manager UI (admin console)
- Redis Stack 7.2 features
- Three Redis database versions: 7.2, 6.2, 6.0
- License file structure updates
- Redis ACL selectors and enhanced key-based permissions
- New INFO fields
- Log rotation enhancements
- Multi-OS upgrade support for clusters with modules

New in this release

New features

Redis 7.0 features

The following Redis 7.0 features are now supported:

- [Redis functions](#)

In Redis Enterprise Software, `FUNCTION STATS` returns an extra parameter, an array called `all_running_scripts`, to reflect multiple functions running at the same time.

- [Multipart AOF](#) (append-only files)
- New commands
- Sharded PUBSUB (see [Sharded pub/sub](#) for details)

Redis 7.2 features

The following Redis 7.2 features are now supported:

- Various performance improvements

- CONFIG SET for locale
- Connection layer modularization
- Encoding improvements: listpack for sets and lists
- Observability: authentication metrics (exposed by INFO command)
- Stream consumer group improvements
- Commands: ZRANK, ZREVRANK new WITHSCORE option
- Shard IDs in cluster shards topology
- Introduce shard ID to Redis cluster
- Support CLIENT NO-TOUCH command
- WAIT AOF

Auto Tiering - Redis on Flash evolution doubles throughput and reduces latency

Redis Enterprise version 7.2 introduces Auto Tiering as an enhanced successor to Redis on Flash, which allows you to provision larger databases at a lower cost by extending the RAM with flash drives.

Redis Enterprise Auto Tiering replaces RocksDB with [Speedb](#) as its storage engine, doubling the throughput and reducing latencies, achieved using the same infrastructure resources. For example, a 1 TB database with 50K ops/sec can now serve 100K ops/sec based on the same infrastructure.

To switch existing databases to use Speedb for Auto Tiering and improve performance:

1. Upgrade the cluster to Redis Enterprise Software version 7.2.4.
2. Upgrade each database with Auto Tiering enabled to Redis database version 7.2.

For more information about Auto Tiering, see:

- [Auto Tiering overview](#)
- [Auto Tiering quick start](#)

RESP3 support

Support for RESP3 and the [HELLO](#) command was added in Redis Enterprise 7.2.

To use RESP3 with Redis Enterprise:

1. Upgrade Redis servers to version 7.2 or later.

For Active-Active and Replica Of databases:

1. Upgrade all participating clusters to Redis Enterprise version 7.2.x or later.
2. Upgrade all databases to version 7.x or later.

2. Enable RESP3 support for your database (enabled by default):

```
rладmin tune db db:<ID> resp3 enabled
```

If you run Redis Stack commands with Redis clients [Go-Redis](#) version 9 or [Lettuce](#) versions 6 and later, see [client prerequisites](#) before you upgrade to Redis 7.2 to learn how to prevent potential application issues due to RESP3 breaking changes.

Sharded pub/sub

[Sharded pub/sub](#) is now supported.

You cannot use sharded pub/sub if you [deactivate RESP3 support](#).

New Cluster Manager UI preview

A preview of the new Cluster Manager UI (admin console) is available in Redis Enterprise Software version 7.2.4.

To try out the new UI:

- On the sign-in screen:
 1. Enter your credentials.
 2. Select **Sign in the new interface**.
- Sign in directly from the new UI's sign-in screen at `https://<hostname or IP address>:8443/new`
- If you are currently signed in to the legacy UI:
 1. Select **Switch to the new Admin Console** to expand the banner at the top of the screen.
 2. Click the **Try it now** button to open the new UI in another tab.

New UI benefits

- User-driven design
- Provides full functionality to complete tasks entirely in the UI
- New attributes and improved feature visibility
- Provides configuration flexibility while highlighting the recommended path
- Addresses the needs of different personas and use cases
- Quicker troubleshooting and easier maintenance

New UI highlights

- More configurable database attributes, including replica high availability, shards placement, and proxy policy.
- Nodes indicate whether it's a primary or secondary node.
- Modules show the databases that are using them.
- Certificates show expiration and validity, and you can upload and copy certificates.
- Cluster license displays the number of shards that are used out of the number of shards that are licensed to the cluster. The new UI allows you to paste or upload a new license.
- Role-based access control (RBAC) has explanations to improve clarity.
- Access Control List (ACLs) now support defining ACLs for modules.
- The databases screen has more information per database for faster troubleshooting. It also allows you to filter databases and compare database metrics.
- The cluster name, user, and user role are shown in the upper right for quickly identifying the cluster from any screen. You can also **Change user password** from the dropdown menu.
- Auto Tiering licensing and a toggle for the storage engine used in Auto Tiering enabled databases (available only in the new UI).
- Input validations.

New UI limitations

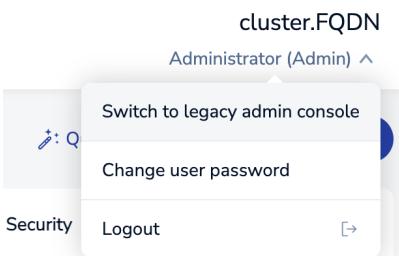
The following features are not supported in this preview but will be added in future releases. Until then, temporarily switch to the legacy UI to do the following:

- Provision and configure Active-Active databases (viewing is available).
- Search and export the event log.
- Remove a node from the UI.

Additional limitations:

- Although Redis supports memcached databases, the new UI only allows view and delete. Memcached users are advised to migrate to Redis to enjoy the full benefits of Redis and its UI.

To open the legacy admin console when signed in to the new UI, select your username, then select **Switch to legacy Admin Console** from the list:



Future UI enhancements

- Configure default database settings and database upgrade settings
- Security preferences related to password and login management
- LDAP improvements
- IPv6 support
- ACL improvements, such as ACLv2 smart validations
- And more

! Note: With the release of the new Cluster Manager UI, the legacy UI is considered deprecated and will eventually be phased out. New functionality will only be implemented in the new Cluster Manager UI, and the old UI will no longer be maintained except for critical bug fixes.

Redis Stack 7.2 features

Redis Enterprise Software version 7.2.4 supports features included in Redis Stack version 7.2.

The following sections include a few highlights. For more details, see the [Redis Stack 7.2 release notes](#).

Search and query

- Introduces [Geo Polygon Search](#). Geo range queries now accept the [GEOSHAPE](#) field type, which supports polygon shapes using [WKT notation](#). GEOSHAPE supports POLYGON and POINT as shape formats and polygon operations.
- Performance improvements for SORT BY operations using [FT . SEARCH](#) and [FT . AGGREGATE](#).
- New FORMAT for improved readability and future support for better error handling responses on FT . SEARCH and FT . AGGREGATE in RESP3 only.

JSON

JSON introduces two new commands:

- [JSON.MERGE](#) merges a given JSON value into matching paths to update, delete, or expand the JSON values at the matching paths.
- [JSON.MSET](#) sets or updates one or more JSON values according to specified key-path-value triplets.

Triggers and functions preview

A preview of triggers and functions is available.

Triggers and functions provide support for running JavaScript functions inside the Redis process. These functions can be executed on-demand, by an event-driven trigger, or by a stream processing trigger.

Try it out with the [triggers and functions quick start](#).

! Note:

- The preview version of triggers and functions is not intended for production use since the API might change in the future and potentially cause application issues when upgrading to a later version.
- During preview, triggers and functions are not supported for databases with Auto Tiering enabled (previously known as Redis on Flash).

Module versions

Redis Enterprise Software version 7.2.4 includes the following Redis Stack modules:

- [Redisearch 2.8.4](#)
- [RedisJSON 2.6.6](#)
- [RedisTimeSeries 1.10.4](#)
- [RedisBloom 2.6.3](#)
- [RedisGears 2.0.11](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Enhancements

Three Redis database versions

Redis Enterprise Software version 6.x includes two Redis database versions: 6.0 and 6.2. As of version 7.2, Redis Enterprise Software includes three Redis database versions: 6.0, 6.2, and 7.2.

To view available Redis database versions:

- In the Cluster Manager UI, see [Redis database versions](#) on the **Cluster > Configuration** screen.
- Send a [GET /nodes REST API request](#) and see `supported_database_versions` in the response.

The default Redis database version, which is used when you upgrade an existing database or create a new one, differs between Redis Enterprise releases as follows:

Redis Enterprise	Bundled Redis DB versions	Default DB version (upgraded/new databases)
7.2	6.0, 6.2, 7.2	7.2
6.4.2	6.0, 6.2	6.2
6.2.x	6.0, 6.2	6.0

For Redis Enterprise Software version 7.2.4, `default_redis_version` is 7.2 for both `major` and `latest` upgrade policies.

Updated Redis Enterprise license format

Redis Enterprise Software version 7.2.4 includes updates to its license format, which add separate shard limits for RAM and flash shards used for Auto Tiering.

For more information, see [Cluster license keys](#).

Redis ACL selectors and key-based permissions

Redis ACLs in Redis Enterprise version 7.2 support key permissions and selectors.

Key permissions:

- `%R~<pattern>`: Grants read access to keys that match the given pattern.
- `%W~<pattern>`: Grants write access to keys that match the given pattern.
- `%RW~<pattern>`: Alias for `~<pattern>`. Grants read and write access to keys that match the given pattern.

See [key permissions](#) for more information.

Selectors let you define multiple sets of rules in a single Redis ACL (only supported for databases with Redis version 7.2 or later). A command is allowed if it matches the base rule or any selector in the Redis ACL. See [selectors](#) for more information.

- `(<rule list>)`: Creates a new selector.
- `clearselectors`: Deletes all existing selectors for a user. This action does not delete the base ACL rule.

Redis ACLs have the following differences in Redis Enterprise Software:

- Nested selectors are not supported.

For example, the following selectors are not valid in Redis Enterprise: +GET ~key1 (+SET (+SET ~key2) ~key3)

- Key and pub/sub patterns do not allow the following characters: ' (', ') '
- The following password syntax is not supported: '>', '<', '#!', 'resetpass'

To change passwords in Redis Enterprise Software, use one of the following methods:

- Cluster Manager UI (admin console)
- `radmin cluster reset_password`:

```
radmin cluster reset_password <user email>
```

- REST API `PUT /v1/users` request and provide password
- The **ACL builder** does not support selectors and key permissions. Use **Free text command** to manually define them instead.

New INFO fields

The **INFO** command includes new fields:

- Under the STATS section:
 - `current_eviction_exceeded_time` - Redis Enterprise reply is always "0"
 - `total_eviction_exceeded_time` - Redis Enterprise reply is always "0"
 - `current_active_defrag_time` - Redis Enterprise reply is always "0"
 - `total_active_defrag_time` - Redis Enterprise reply is always "0"
- Under the MEMORY section:
 - `maxmemory_policy` - The value of the `maxmemory-policy` configuration directive

The **INFO** command can now accept multiple section arguments (requires Redis database version 7 or later).

Log rotation enhancements

- The `logrotate` tool rotates logs that exceed 200 MB.
- `logrotate` runs every five minutes instead of once a day.
- The job scheduler runs `logrotate` instead of the OS.
- Every cluster upgrade overwrites the log rotation configuration.
- You can edit the log rotation configuration at `$pkgconfdir/logrotate.conf` (`pkgconfdir` is `/opt/redislabs/config` by default, but can be changed in a custom installation). Note that the configuration file moved since last version.
- You can change how often the `logrotate` tool runs using the job scheduler REST API request `PUT /v1/job_scheduler`.

Multi-OS upgrade support for clusters with modules

Starting from Redis Enterprise version 7.2, all future 7.2.x upgrades are supported for clusters containing databases with modules in combination with with Operating System (OS) upgrades.

Resolved issues

- RS54131 - +OK reply not received on TLS-enabled database
- RS101525 - Cluster provides wrong number of database connections on Grafana
- RS104028 - Fix the self-signed certificate script: error generating certificates with multiple FQDNs
- RS87920 - Proxy log is full of the warning message "Failed to check status of running child syncer process 0 : No child processes"
- RS99916 - Fixed the UI log to include the names of LDAP users at login
- RS84273 - When an LDAP user with a Redis admin role viewed the log in the UI, they received `db_viewer` permissions instead of `admin`, which

limited log visibility

Version changes

Breaking changes

- Differences when using the UNWATCH command within a MULTI command sequence:
 - Redis Enterprise: UNWATCH is not allowed within a MULTI command sequence and returns an error.
 - OSS: UNWATCH is allowed within a MULTI sequence but has no effect.
- When sending a PUBSUB SHARDNUMSUB command in OSS Cluster mode in Redis Enterprise, Redis Enterprise checks the hash slots of the requested channels. Redis Enterprise responds with a CROSSSLOT error if the channels don't hash to the same slot, or aMOVED error if the channels hash to a different node.

Redis 7.2 breaking changes

When new major versions of open source Redis change existing commands, upgrading your database to a new version can potentially break some functionality. Before you upgrade, make sure to read the provided list of breaking changes that affect Redis Enterprise and update any applications that connect to your database to handle these changes.

Confirm your Redis database version (`redis_version`) using the admin console or run the following `INFO` command via `redis-cli`:

```
$ redis-cli -p <port> INFO
"# Server
redis_version:7.0.8
..."
```

Breaking changes from version 6.2

Upgrading to open source Redis version 7.2 from version 6.2 introduces the following potentially breaking changes to Redis Enterprise.

Programmability

- Lua scripts no longer have access to the `print()` function ([#10651](#)) - The `print` function was removed from Lua because it can potentially cause the Redis processes to get stuck (if no one reads from stdout). Users should use `redis.log`. An alternative is to override the `print` implementation and print the message to the log file.
- Block `PFCOUNT` and `PUBLISH` in read-only scripts (*_RO commands, no-writes) ([#10744](#)) - Consider `PFCOUNT` and `PUBLISH` as write commands in scripts, in addition to `EVAL`; meaning:
 - They can never be used in scripts with shebang (#!) and no no-writes flag
 - They are blocked in `EVAL_RO` and `_RO` variants, (even in scripts without shebang (#!) flags)
 - Allow no-write scripts in `EVAL` (not just in `EVAL_RO`), even during `CLIENT PAUSE WRITE`
- Hide the `may_replicate` flag from the `COMMAND` command response ([#10744](#)) - As part of the change to treat `may_replicate` commands `PFCOUNT` and `PUBLISH` as write commands in scripts, in addition to `EVAL`, the `may_replicate` flag has been removed from the `COMMAND` response.
- Time sampling is now frozen during command execution and scripts ([#10300](#)). While a command or script is running, the keys used by the command or script will not expire. This breaks any script that uses a loop to wait for a key to expire.
- Blocked commands in scripts now work the same way as when they are used in transactions ([#11568](#)).

Error handling

- Rephrased some error responses about invalid commands or arguments ([#10612](#)) -
 - Error response for unknown command introduced a case change (Unknown to unknown)
 - Errors for module commands extended to cover subcommands, updated syntax to match Redis Server syntax
 - Arity errors for module commands introduce a case change (Wrong to wrong); will consider full command name
- Corrected error codes returned from `EVAL` scripts ([#10218](#), [#10329](#)).

These examples show changes in behavior:

```

1: config set maxmemory 1
2: +OK
3: eval "return redis.call('set','x','y')" 0
- 4: -ERR Error running script (call to 71e6319f97b0fe8bdfa1c5df3ce4489946dda479): @user_script:1:
@user_script: 1: -00M command not allowed when used memory > 'maxmemory'.
+ 4: -ERR Error running script (call to 71e6319f97b0fe8bdfa1c5df3ce4489946dda479): @user_script:1: 00M
command not allowed when used memory > 'maxmemory'.
5: eval "return redis.pcall('set','x','y')" 0
- 6: -@user_script: 1: -00M command not allowed when used memory > 'maxmemory'.
+ 6: -00M command not allowed when used memory > 'maxmemory'.
7: eval "return redis.call('select',99)" 0
8: -ERR Error running script (call to 4ad5abfc50bbccb484223905f9a16f09cd043ba8): @user_script:1: ERR DB
index is out of range
9: eval "return redis.pcall('select',99)" 0
10: -ERR DB index is out of range
11: eval_ro "return redis.call('set','x','y')" 0
-12: -ERR Error running script (call to 71e6319f97b0fe8bdfa1c5df3ce4489946dda479): @user_script:1:
@user_script: 1: Write commands are not allowed from read-only scripts.
+12: -ERR Error running script (call to 71e6319f97b0fe8bdfa1c5df3ce4489946dda479): @user_script:1: ERR
Write commands are not allowed from read-only scripts.
13: eval_ro "return redis.pcall('set','x','y')" 0
-14: -@user_script: 1: Write commands are not allowed from read-only scripts.
+14: -ERR Write commands are not allowed from read-only scripts.

```

- [ZPOPMIN/ZPOPMAX](#) used to produce wrong replies when count is 0 with non-zset [#9711](#):

- ZPOPMIN/ZPOPMAX used to produce an (empty array) when key was not a sorted set and the optional count argument was set to 0 and now produces a WRONGTYPE error response instead.
- The optional count argument must be positive. A negative value produces a value is out of range error.

These examples show changes in behavior:

```

1: zadd myzset 1 "one"
2: (integer) 1
3: zadd myzset 2 "two"
4: (integer) 1
5: zadd myzset 3 "three"
6: (integer) 1
7: zpopmin myzset -1
- 8: (empty array)
+ 8: (error) ERR value is out of range, must be positive
9: 127.0.0.1:6379> set foo bar
10: OK
11: zpopmin foo 0
-12: (empty array)
+12: (error) WRONGTYPE Operation against a key holding the wrong kind of value

```

- [LPOP/RPOP](#) with count against a nonexistent list returns a null array instead of (nil) [\(#10095\)](#). This change was backported to 6.2.

- [LPOP/RPOP](#) used to produce (nil) when count is 0, now produces a null array [\(#9692\)](#). This change was backported to 6.2.

- [XCLAIM/XAUTOCLAIM](#) skips deleted entries instead of replying with nil and deletes them from the pending entry list [\(#10227\)](#) - XCLAIM/XAUTOCLAIM now behaves in the following way:

- If you try to claim a deleted entry, it is deleted from the pending entry list (PEL) where it is found (as well as the group PEL). Therefore, such an entry is not claimed, just cleared from PEL (because it doesn't exist in the stream anyway).
 - Because deleted entries are not claimed, X[AUTO]CLAIM does not return "nil" instead of an entry.
 - Added an array of all the deleted stream IDs to XAUTOCLAIM response.
- A blocked stream command that is released when a key no longer exists returns a different error code [\(#11012\)](#).
 - For newly unblocked streams, lists, and zsets, the old implementation returned UNBLOCKED when the stream key was deleted or overwritten with a different type. Now, errors will be the same as if the command was processed after the effect.
- ACL errors have been unified across Redis. [\(#11160\)](#)
 - When using [RedisModule_Call](#) module API function, ACL errors return -NOPERM instead of -ERR
 - [XREADGROUP](#) and [XAUTOCLAIM](#) create a consumer regardless of whether it was able to perform reading or claiming [\(#11012\)](#).

- Any float that is Not a Number will return nan ([#11597](#)).

ACLs

- [ACL GETUSER](#) reply now uses ACL syntax for keys and channels ([#9974](#)). [ACL GETUSER](#) now uses the ACL DSL (Domain Specific Language) for keys and channels.

These examples show changes in behavior:

```

1: acl setuser foo off resetchannels &channel1 -@all +get
2: OK
3: acl getuser foo
4: 1) "flags"
5: 2) 1) "off"
6: 3) "passwords"
7: 4) (empty array)
8: 5) "commands"
9: 6) "-@all +get"
10: 7) "keys"
-11: 8) (empty array)
+11: 8) ""
12: 9)"channels"
-13 10) 1) "channel1"
+13 10) "&channel1"
```

- [SORT/SORT_RO](#) commands reject key access patterns in GET and BY if the ACL doesn't grant the command full keyspace access ([#10340](#)) - The sort and sort_ro commands can access external keys via GET and BY. In order to make sure the user cannot violate the authorization ACL rules, Redis 7 will reject external keys access patterns unless ACL allows SORT full access to all keys. For backwards compatibility, SORT with GET/BY keeps working, but if ACL has restrictions to certain keys, the use of these features will result in a permission denied error.

These examples show changes in behavior:

```

USER FOO (+sort ~* ~mylist)
#F00> sort mylist by w* get v* - is O.K since ~* provides full key access

USER FOO (+sort %R~* ~mylist)
#F00> sort mylist by w* get v* - is O.K since %R~* provides full key READ access**

USER FOO (+sort %W~* ~mylist)
#F00> sort mylist by w* get v* - will now fail since $W~* only provides full key WRITE access

USER FOO (+sort ~v* ~mylist)
#F00> sort mylist by w* get v* - will now fail since ~v* only provides partial key access
```

- Fix ACL category for [SELECT, WAIT, ROLE, LASTSAVE, READONLY, READWRITE, ASKING](#) ([#9208](#)):

- SELECT and WAIT have been recategorized from @keyspace to @connection
- ROLE, LASTSAVE have been categorized as @admin and @dangerous
- ASKING, READONLY, READWRITE have also been assigned the @connection category and removed from @keyspace
- Command categories are explained in [ACL documentation](#)
- When a blocked client is being unblocked, checks for ACLs and OOM condition checks are now re-evaluated ([#11012](#)).
 - If the ACL rules have changed since the command was executed, the command might fail after the client is unblocked.

Command introspection, stats, and configuration

- [COMMAND](#) reply drops random and sort-for-scripts flags, which are now part of [command tips](#) ([#10104](#)) - The random flag was replaced with the nondeterministic_output tip; the sort-for-scripts flag was replaced by the nondeterministic_order tip
- [INFO commandstats](#) now shows the stats per sub-command ([#9504](#)) For example, while previous versions would provide a single entry for all command usage, in Redis 7, each sub command is reported separately:
 - Redis 6.2:

```
cmdstat_acl:calls=4,usec=279,usec_per_call=69.75,rejected_calls=0,failed_calls=2
```

- Redis 7:

```
cmdstat_acl|list:calls=1,usec=4994,usec_per_call=4994.00,rejected_calls=0,failed_calls=0  
cmdstat_acl|setuser:calls=2,usec=16409,usec_per_call=8204.50,rejected_calls=0,failed_calls=0  
cmdstat_acl|deluser:calls=1,usec=774,usec_per_call=774.00,rejected_calls=0,failed_calls=0  
cmdstat_acl|getuser:calls=1,usec=6044,usec_per_call=6044.00,rejected_calls=0,failed_calls=0
```

- [CONFIG REWRITE](#), [CONFIG RESETSTAT](#), and most [CONFIG SET](#) commands are now allowed during loading (#9878)
- When running [XINFO CONSUMERS](#), the `idle` time now shows the number of milliseconds that have passed since the last attempted interaction, and the `inactive` time shows the number of milliseconds since the last successful interaction (#11099)
 - Previously, `idle` time showed the number of milliseconds that passed since the last successful interaction and there was no `inactive` time.
- Command stats are only updated when the command executes (#11012).
 - Previously, the command stats were updated even if a command was blocked. The command stats are now updated only if and when the command is executed.

Client prerequisites for Redis 7.2 upgrade

The Redis clients [Go-Redis](#) version 9 and [Lettuce](#) versions 6 and later use RESP3 by default. If you use either client to run Redis Stack commands, you should set the client's protocol version to RESP2 before upgrading your database to Redis version 7.2 to prevent potential application issues due to RESP3 breaking changes.

For applications using Go-Redis v9.0.5 or later, set the protocol version to RESP2:

```
client := redis.NewClient(&redis.Options{  
    Addr:     "<database_endpoint>",  
    Protocol: 2, // Pin the protocol version  
})
```

To set the protocol version to RESP2 with Lettuce v6 or later:

```
import io.lettuce.core.*;  
import io.lettuce.core.api.*;  
import io.lettuce.core.protocol.ProtocolVersion;  
  
// ...  
RedisClient client = RedisClient.create("<database_endpoint>");  
client.setOptions(ClientOptions.builder()  
    .protocolVersion(ProtocolVersion.RESP2) // Pin the protocol version  
    .build());  
// ...
```

If you are using [LettuceMod](#), you need to upgrade to [v3.6.0](#).

Deprecations

Command deprecations

- [CLUSTER SLOTS](#) is deprecated as of Redis 7.0
- [JSON RESP](#) is deprecated as of Redis Stack 7.2.
- [QUIT](#) is deprecated as of Redis 7.2

API deprecations

Fields deprecated as of Redis Enterprise v4.3.3:

- `smtp_use_tls` (replaced with `smtp_tls_mode`)
- `dns_address_master`

- endpoint_node
- endpoint_ip
- public_addr (replaced with external_addr)

Fields deprecated as of Redis Enterprise v4.4.2:

- default_shards_overbooking (replaced with shards_overbooking)

Fields deprecated as of Redis Enterprise v6.4.2:

- use_ipv6 (replaced with use_external_ipv6)
- redis_cleanup_job_settings (replaced with persistence_cleanup_scan_interval)

Fields deprecated as of Redis Enterprise v5.0.1:

- bdb_high_syncer_lag (replaced with replica_src_high_syncer_lag and crdt_src_high_syncer_lag)
- bdb_syncer_connection_error
- bdb_syncer_general_error
- sync_sources (replaced with replica_sources and crdt_sources)
- sync (replaced with replica_sync and crdt_sync)
- ssl (replaced with tls_mode)

Fields deprecated as of Redis Enterprise v7.2:

- node.bigstore_driver (replaced with cluster.bigstore_driver)
- auth_method
- authentication_redis_pass (replaced with multiple passwords feature in version 6.0.X)

Other deprecated fields:

- import/rdb_url (deprecated as of Redis Enterprise v4.X)
- logrotate_dir (to be replaced with logrotate_config or removed)

Deprecated CLI commands:

- rlutil change_master (deprecated as of Redis Enterprise v6.2.18, replaced with rladmin change_master)
- rlutil reserved_ports (deprecated as of Redis Enterprise v7.2, replaced with rladmin cluster config reserved_ports)

REST API requests deprecated as of Redis Enterprise v7.2:

- POST /v1/modules (replaced with POST /v2/modules)
- DELETE /v1/modules (replaced with DELETE /v2/modules)

Access control deprecations

- The following predefined roles and Redis ACLs are no longer available for new Redis Enterprise Software version 7.2.4 clusters:
 - Custom roles (not management roles): Cluster Member, Cluster Viewer, DB Member, DB Viewer, None.
 - Redis ACLs: Not Dangerous and Read Only.
- In upcoming maintenance releases, the deprecated roles and ACLs will be removed automatically when you upgrade to Redis Enterprise Software version 7.2.4, unless they are associated with any users or databases in the cluster.
- A deprecation notice for SASL-based LDAP was included in [previous Redis Enterprise Software release notes](#). When you upgrade to Redis Enterprise Software version 7.2.4, all existing “external” users (previously used to support SASL-based LDAP) will be removed.

Legacy UI

With the release of the new Cluster Manager UI, the legacy UI is considered deprecated and will eventually be phased out. New functionality will only be

implemented in the new Cluster Manager UI, and the old UI will no longer be maintained except for critical bug fixes.

RedisGraph

Redis has announced the end of life for RedisGraph. Redis will continue to support all RedisGraph customers, including releasing patch versions until January 31, 2025.

See the [RedisGraph end-of-life announcement](#) for more details.

RHEL and CentOS 7.0-7.9

Support for RHEL and CentOS 7.0-7.9 is considered deprecated and will be removed in a future release.

Oracle Linux 7

Oracle Linux 7 support is considered deprecated and will be removed in a future release.

Amazon Linux 1

Amazon Linux 1 support is considered deprecated and will be removed in a future release.

Ubuntu 16.04

The deprecation of Ubuntu 16.04 was announced in the [Redis Enterprise Software 6.4.2 release notes](#). As of Redis Enterprise Software 7.2.4, Ubuntu 16.04 is no longer supported.

RC4 encryption cipher

The RC4 encryption cipher is considered deprecated in favor of stronger ciphers. Support for RC4 by the [discovery service](#) will be removed in a future release.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES. Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later. Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

Upcoming changes

Prepare for restrictive pub/sub permissions

Redis database version 6.2 introduced pub/sub ACL rules that determine which [pub/sub channels](#) a user can access.

The configuration option `acl-pubsub-default`, added in Redis Enterprise Software version 6.4.2, determines the cluster-wide default level of access for all pub/sub channels. Redis Enterprise Software uses the following pub/sub permissions by default:

- For versions 6.4.2 and 7.2, `acl-pubsub-default` is permissive (`allchannels` or `&*`) by default to accommodate earlier Redis versions.
- In future versions, `acl-pubsub-default` will change to restrictive (`resetchannels`). Restrictive permissions block all pub/sub channels by default, unless explicitly permitted by an ACL rule.

If you use ACLs and pub/sub channels, you should review your databases and ACL settings and plan to transition your cluster to restrictive pub/sub permissions in preparation for future Redis Enterprise Software releases.

When you change the cluster's default pub/sub permissions to restrictive, `&*` is added to the **Full Access** ACL. Before you make this change, consider the following:

- Because pub/sub ACL syntax was added in Redis 6.2, you can't associate the **Full Access** ACL with database versions 6.0 or lower after this change.
- The **Full Access** ACL is not reverted if you change `acl-pubsub-default` to permissive again.
- Every database with the default user enabled uses the **Full Access** ACL.

To secure pub/sub channels and prepare your cluster for future Redis Enterprise Software releases that default to restrictive pub/sub permissions:

1. Upgrade Redis databases:

- For Redis Enterprise Software version 6.4.2, upgrade all databases in the cluster to Redis DB version 6.2.
- For Redis Enterprise Software version 7.2.4, upgrade all databases in the cluster to Redis DB version 7.2 or 6.2.

2. Create or update ACLs with permissions for specific channels using the `resetchannels &channel` format.

3. Associate the ACLs with relevant databases.

4. Set default pub/sub permissions (`acl-pubsub-default`) to restrictive. See [Change default pub/sub permissions](#) for details.

5. If any issues occur, you can temporarily change the default pub/sub setting back to permissive. Resolve any problematic ACLs before making pub/sub permissions restrictive again.

Upcoming command request and response changes

Open source Redis version 7.2 changes the request and response policies for several commands. Because the GA release of Redis Enterprise version 7.2 does not include these policy changes, commands might behave differently from open source Redis. However, these changes will be included in a future Redis Enterprise maintenance release:

- `RANDOMKEY` and `SCAN` will change from no response policy to a SPECIAL response policy.
- `MSETNX` currently has a MULTI_SHARD request policy and AGG_MIN response policy. Both will change to no policy.

For more information about request and response policies, see [Redis command tips](#).

Supported platforms

 Supported – The platform is supported for this version of Redis Enterprise Software.

 Deprecated – The platform is still supported for this version of Redis Enterprise Software, but support will be removed in a future release.

 End of life – Platform support ended in this version of Redis Enterprise Software.

Redis Enterprise	7.2.4	6.4.2	6.2.18	6.2.12	6.2.10	6.2.8	6.2.4
Ubuntu¹							
20.04		 ⁶	–	–	–	–	–
18.04							
16.04							
RHEL & CentOS²							
8.8		–	–	–	–	–	–
8.7			–	–	–	–	–
8.5-8.6						–	–
8.0-8.4							–
7.0-7.9							
Oracle Linux³							
8						–	–
7							
Rocky Linux³							
8				–	–	–	–
Amazon Linux							
2		 ⁷	–	–	–	–	–
1							
Docker⁴							
Kubernetes ⁵							

1. The server version of Ubuntu is recommended for production installations. The desktop version is only recommended for development deployments.

2. RHEL and CentOS deployments require OpenSSL 1.0.2 and [firewall configuration](#).

3. Based on the corresponding RHEL version.
4. Docker images of Redis Enterprise Software are certified for development and testing only.
5. See the [Redis Enterprise for Kubernetes documentation](#).
6. Ubuntu 20.04 support was added in Redis Enterprise Software [6.4.2-43](#).
7. A release candidate for Amazon Linux 2 support was added in Redis Enterprise Software [6.4.2-61](#). Official support for Amazon Linux 2 was added in Redis Enterprise Software [6.4.2-69](#).

Downloads

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (7.2.4-52 August release)
Ubuntu 18	7c7e465c8e129a03ee9f585137b2a1d9
Ubuntu 20	631f27311b19806955fde012953ff9c9
RedHat Enterprise Linux (RHEL) 7	ae76798b1b7243313b4f4cba6ede88d7
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	48936b25aefa2921d38aea84ad06134d
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	
Amazon Linux 2	3e8180d7a9ebc3784ab6080234edefd5

Known issues

Legacy UI known issues

When using the legacy UI, you cannot update and save your changes on the `settings > preferences` tab even though these settings are visible. This issue will be fixed in the next maintenance release.

As a workaround, use the new Cluster Manager UI to update these settings from the `Cluster > Security > Preferences` tab.

Pub/sub channel ACL limitations

In Redis Enterprise Software version 6.4.2, you could use `&channel` syntax in Redis ACL rules to allow access to specific pub/sub channels even when default pub/sub permissions were permissive (`&allchannels` or `&*`), allowing all channels by default. However, `&allchannels &channel` is not valid syntax.

As of Redis Enterprise Software version 7.2.4, you cannot create Redis ACLs with this combination of rules. You can only use `&channel` to allow access to specific channels if the default pub/sub permissions are restrictive (`resetchannels`).

Associating an ACL that contains the invalid syntax `&allchannels &channel` (created before version 7.2) with a user and database might leave the database in a pending state, unable to function.

To prevent this issue:

1. Review all existing ACL rules.
2. For each rule containing `&channel`, either:
 - Add the `resetchannels` prefix to restrict access to all channels by default.
 - Delete the rule if not needed.

Known limitations

Command limitations

- `CLIENT NO-TOUCH` might not run correctly in the following cases:
 - The Redis database version is earlier than 7.2.0.
 - The `CLIENT NO-TOUCH` command is forbidden by ACL rules.

Before sending this command, clients should verify the database version is 7.2.0 or later and that using this command is allowed.

- You cannot use [SUNSUBSCRIBE](#) to unsubscribe from a shard channel if the regex changed while subscribed.
- Using [XREADGROUP BLOCK](#) with > to return all new streams will cause the Redis database to freeze until the shard is restarted. ([#12031](#))
- Because a rejected command does not record the duration for command stats, an error will appear after it is reprocessed that will cause the Redis database to freeze until the shard is restarted. ([#12247](#))

Modules cannot load in Oracle Linux 7 & 8

Databases hosted on Oracle Linux 7 & 8 cannot load modules.

As a temporary workaround, you can change the node's os_name in the Cluster Configuration Store (CCS):

```
ccs-cli hset node:<ID> os_name rhel
```

Cluster recovery with manually uploaded modules

For clusters containing databases with manually uploaded modules, [cluster recovery](#) requires an extra step.

After installing Redis Enterprise Software on the cluster nodes, upload compatible modules to `modulesdir` (/opt/redislabs/lib/modules) before continuing the recovery process.

This limitation will be removed in a future maintenance release.

Security

Open source Redis security fixes compatibility

As part of Redis's commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with [open source Redis](#). The following open source Redis [CVEs](#) do not affect Redis Enterprise:

- [CVE-2021-32625](#) — Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) — Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32675](#) — Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) — Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) — Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

Redis Enterprise has already included the fixes for the relevant CVEs. Some CVEs announced for open source Redis do not affect Redis Enterprise due to different and additional functionality available in Redis Enterprise that is not available in open source Redis.

Updated: August 31, 2023

Redis Enterprise Software release notes 6.4.2

[Redis Enterprise Software version 6.4.2](#) is now available!

This version offers:

- Extended validation of client certificates via mTLS (mutual TLS) full subject support
- Support for default restrictive permissions when using publish/subscribe commands and access control lists (ACLs)
- Enhanced TLS performance when Redis returns large arrays in responses
- Compatibility with [open source Redis](#) 6.2.7 and 6.2.10.

- Additional enhancements and bug fixes

Detailed release notes

For more detailed release notes, select a build version from the following table:

Version (Release date)	Major changes	OSS Redis compatibility
6.4.2-94 (July 2023)	Look-ahead mechanism for planner attempts. Package OS compatibility validation.	Redis 6.2.10
6.4.2-81 (June 2023)	Email alerts for database backup failures and replica high availability shard relocation failures.	Redis 6.2.10
6.4.2-69 (May 2023)	Amazon Linux 2 support. Configure envoy ports using rladmin. Added option to avoid specific nodes when using the optimized shards placement API. Added failure_detection_sensitivity to replace watchdog_profile.	Redis 6.2.10
6.4.2-61 (April 2023)	Amazon Linux 2 support. Fixed known limitations for custom installation on RHEL 7 and RHEL 8, running rl_rdbconvert manually, and resharding rack-aware databases with no replication.	Redis 6.2.10
6.4.2-43 (March 2023)	Ubuntu 20.04 support. Safe node removal. Allow gossip_envoy port configuration.	Redis 6.2.10
6.4.2-30 (February 2023)	Pub/sub ACLs & default permissions. Validate client certificates by subject attributes.	Redis 6.2.7

Deprecations

Ubuntu 16.04

Ubuntu 16 support is considered deprecated and will be removed in a future release. Ubuntu 16.04 LTS (Xenial) has reached the end of its free initial five-year security maintenance period as of April 30, 2021.

Active-Active database persistence

The RDB snapshot option for [Active-Active database persistence](#) is deprecated and will be removed in a future release.

Please plan to reconfigure any Active-Active databases to use append-only file (AOF) persistence with the following command:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> \
--default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later. Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES. Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

Known limitations

Feature limitations

- RS97971 - [Resharding fails for rack-aware databases with no replication](#) (fixed and resolved as part of [v6.4.2-61](#)).
- RS101204 - High memory consumption caused by the `persistence_mgr` service when AOF persistence is configured for every second (fixed and resolved as part of [v6.4.2-81](#)).
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use

`rladmin cluster config` to configure handle_redirects or handle_metrics_redirects.

- RS51144, RS102128 - Active-Active: To start successfully, the syncer (crdt-syncer) must connect to all sources. In multi-cluster configurations (more than 2 A-A clusters participating), in some cases, if one or more of the clusters is not available, A-A replication will be down.

Resharding fails for rack-aware databases with no replication

When a database is configured as [rack-aware](#) and replication is turned off, the resharding operation fails.

RS97971 - This limitation was fixed and resolved as part of [v6.4.2-61](#).

Workaround:

Before resharding your database, turn off rack awareness:

```
curl -k -u "<user>:<password>" -H "Content-type: application/json" -d '{"rack_aware": false}' -X PUT  
"https://localhost:9443/v1/b dbs/<bdb_uid>"
```

After the resharding process is complete, you can re-enable rack awareness:

```
curl -k -u "<user>:<password>" -H "Content-type: application/json" -d '{"rack_aware": true}' -X PUT  
"https://localhost:9443/v1/b dbs/<bdb_uid>"
```

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory /etc/opt/redislabs/ even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.4.2-30, perform the following steps:

1. Use crdb-cli to verify that the modules ([modules](#)) and their versions (in [module_list](#)) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's [redis modules](#) tab, validate that these modules with their specific versions are loaded to the cluster.

3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

This limitation has been fixed and resolved as of [v6.4.2-43](#).

Operating system limitations

RHEL 7 and RHEL 8

RS95344 - CRDB database will not start on Redis Enterprise v6.4.2 with a custom installation path.

For a workaround, use the following commands to add the relevant CRDB files to the Redis library:

```
$ yum install -y chrpath  
$ find $installdir -name "crdt.so" | xargs -n1 -I {} /bin/bash -c 'chrpath -r ${libdir} {}'
```

This limitation has been fixed and resolved as of [v6.4.2-61](#).

RHEL 8

Due to module binary differences between RHEL 7 and RHEL 8, you cannot upgrade RHEL 7 clusters to RHEL 8 when they host databases using modules. Instead, you need to create a new cluster on RHEL 8 and then migrate existing data from your RHEL 7 cluster. This does not apply to clusters that do not use modules.

Ubuntu 20.04

By default, you cannot use the SHA1 hash algorithm ([OpenSSL's default security level is set to 2](#)). The operating system will reject SHA1 certificates even if the `mtls_allow_weak_hashing` option is enabled. You need to replace SHA1 certificates with newer certificates that use SHA-256. Note that the certificates provided with Redis Enterprise Software use SHA-256.

Modules not supported for Amazon Linux 2 release candidate

A database with modules cannot reside on an Amazon Linux 2 (release candidate) node. Support was added as part of [v6.4.2-69](#).

Updated: August 17, 2023

Redis Enterprise Software release notes 6.4.2-94 (July 2023)

This is a maintenance release for [Redis Enterprise Software version 6.4.2](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-94 July release)
Ubuntu 16	e664a93108aeb72cf9dc849b84b79239
Ubuntu 18	9b547be7e093a0668caa9d5af36ddd21
Ubuntu 20	d623c8458a28f09a1e6ae0c0a8650023
RedHat Enterprise Linux (RHEL) 7	f9df1080427660bab601f1690ee7ffca
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	f44dfaf4da5d54d35d739ba1e243eca0
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	
Amazon Linux 2	f2c5c2b5e879b8fd6792336f757593e2

New features and enhancements

- RS98361 - Enhanced the algorithm responsible for placing shards in the cluster. This improvement applies when creating new databases and when calling the REST API `GET / optimize_shards_placement` for existing databases.

With this update, the algorithm efficiently finds a legal placement while reducing the calculation time significantly. These changes ensure a smoother and faster experience when managing databases with a large number of shards and dense shard placement policy.

- RS62197 - Added validation of the operating system (OS) version before installation. The installer automatically validates the OS type and version and verifies the correct binary is used for the corresponding system. This prevents missing dependencies which may cause issues later on.

The validation applies to major versions of all operating systems. If the check fails, the installation process halts with an appropriate error message.

Redis Stack v6.2.6

Redis Enterprise Software v6.4.2 includes the new features delivered in the latest [Redis Stack release \(6.2.6 v7\)](#):

- [RediSearch v2.6.9](#)
- [RedisJSON v2.4.7](#)
- [RedisBloom v2.4.5](#)
- [RedisGraph v2.10.10](#)
- [RedisTimeSeries v1.8.10](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Resolved issues

- RS100903 - Fixed a v6.4.2 issue: when a database client requests `SLOWLOG GET [count]`, the proxy ignores “count” and always replies with 128 records
- RS95394 - Fixed Replica Of sync causing high DMC memory usage on source cluster when TLS is enabled “for Replica Of Only”.
- RS102905 - Fixed invalid character that could potentially cause failure in generating support package for Active-Active databases.
- RS92430 - Optimized internal certification rotation process benefiting mainly clusters.
- RS99495 - Updated UI title property to “Redis Enterprise”

Updated: August 17, 2023

Redis Enterprise Software release notes 6.4.2-81 (June 2023)

This is a maintenance release for [Redis Enterprise Software version 6.4.2](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-81 June release)
Ubuntu 16	31631248672de0154ec20aee0bff9adc
Ubuntu 18	eb4e8c1bc1443923ebcd457e51684f7e
Ubuntu 20	1256292b8dca6456e69dc309be51a8d0
RedHat Enterprise Linux (RHEL) 7	1e51c5fdf22c0b6988565444ba08c6cd
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	96dc036339fe18fb218e8d4c8ff50c99
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	
Amazon Linux 2	1a549bdb360c9e22ef764faeadbd4c90

New features and enhancements

- RS77279 - Database backup failures and replica high availability ([replica HA](#)) failures (for example, when relocating a shard from a node) now generate email alerts

Redis Stack v6.2.6

Redis Enterprise Software v6.4.2 includes the new features delivered in the latest [Redis Stack release \(6.2.6 v6\)](#):

- [RediSearch v2.6.9](#)
- [RedisJSON v2.4.7](#)
- [RedisBloom v2.4.5](#)
- [RedisGraph v2.10.9](#)
- [RedisTimeSeries v1.8.10](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Resolved issues

- RS101204 - Fixed high memory consumption that was caused by the `persistence_mgr` service when AOF persistence is configured for every second.
- RS39322, RS35526 - `r1check` no longer requires root privileges and can run with the `redislabs` group and user.
- RS97315, RS100853 - Fixed the robustness of the support package process when collecting the shard configuration.
- RS65110 - Fixed firewalld SELinux configuration.

- RS101691 - Clean false error messages at the end of installation.

Known limitations

Feature limitations

- RS78430 - When [tuning module arguments](#), typos or unsupported arguments can cause shards to get stuck after restarting.
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use `radmin cluster config` to configure `handle_redirects` or `handle_metrics_redirects`.

Operating system limitations

Ubuntu 20.04

By default, you cannot use the SHA1 hash algorithm ([OpenSSL's default security level is set to 2](#)). The operating system will reject SHA1 certificates even if the `mtls_allow_weak_hashing` option is enabled. You need to replace SHA1 certificates with newer certificates that use SHA-256. Note that the certificates provided with Redis Enterprise Software use SHA-256.

Updated: August 17, 2023

Redis Enterprise Software release notes 6.4.2-69 (May 2023)

This is a maintenance release for [Redis Enterprise Software version 6.4.2](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-69 May release)
Ubuntu 16	31631248672de0154ec20aee0bff9adc
Ubuntu 18	7d9ebd4ea5a23137d2bb3928134ea6ef
Ubuntu 20	357ceda7467fd66962f3cb5aa143def7
RedHat Enterprise Linux (RHEL) 7	4d53fcce8c80910a05ed37ab64b57a7a
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	88dc6a4d96b948b086f785dc42786d03
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	
Amazon Linux 2	7ed06bc0b35494e4a5c084528d6e8b9f

New features and enhancements

- Amazon Linux 2 support
- Enhanced installation logs (RS69079, RS69571, RS77857, RS89760)
- Added support for `radmin` envoy port configuration (RS95483)
- Support the option to avoid specific nodes when using optimized shards placement API (RS98795)
- Check port availability before installation and upgrade (RS80815)
- Added `failure_detection_sensitivity` policy to [cluster policy REST API requests](#). See the [watchdog_profile deprecation notice](#) for more details.

Redis Stack v6.2.6

Redis Enterprise Software v6.4.2 includes the new features delivered in the latest [Redis Stack release \(6.2.6 v6\)](#):

- [RediSearch v2.6.6](#)

- RedisJSON v2.4.7
- RedisBloom v2.4.5
- RedisGraph v2.10.9
- RedisTimeSeries v1.8.9

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Version changes

Deprecations

watchdog_profile

The `watchdog_profile` setting is deprecated and will be removed in a future release. Instead, use the `failure_detection_sensitivity` policy for predefined thresholds and timeouts:

```
radmin tune cluster failure_detection_sensitivity [ high | low ]
```

The `failure_detection_sensitivity` policy has the following options:

- `high` (previously known as `local-network watchdog_profile`) – high failure detection sensitivity, lower thresholds, and faster failure detection and failover
- `low` (previously known as `cloud watchdog_profile`) – low failure detection sensitivity and higher tolerance for latency variance (also called network jitter)

Ubuntu 16.04

Ubuntu 16 support is deprecated, and support will be removed in a future release.

Active-Active database persistence

The RDB snapshot option for [Active-Active database persistence](#) is deprecated and will be removed in a future release.

Please plan to reconfigure any Active-Active databases to use append-only file (AOF) persistence with the following command:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> \
    --default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

Resolved issues

- RS97528 - Prevent [self-signed certificate](#) script error on Ubuntu 20.04 by changing the user instructions.
- RS99643 - Fix [cipher suites](#) configuration to allow default cipher suites according to the [control plane](#), [data plane](#), and [discovery service](#).
- RS99696 - Fix [HINCRBYFLOAT](#) CVE
- RS54226 - During upgrades, a CRDB worker process would intermittently get stuck in a restart loop. This issue is now fixed.

Known limitations

Feature limitations

- RS78430 - When [tuning module arguments](#), any typo or use of unsupported arguments can cause shards to get stuck after restart.

Operating system limitations

Ubuntu 20.04

By default, you cannot use the SHA1 hash algorithm ([OpenSSL's default security level is set to 2](#)). The operating system will reject SHA1 certificates even if the `mtls_allow_weak_hashing` option is enabled. You need to replace SHA1 certificates with newer certificates that use SHA-256. Note that the certificates provided with Redis Enterprise Software use SHA-256.

Redis Enterprise Software release notes 6.4.2-61 (April 2023)

This is a maintenance release for [Redis Enterprise Software version 6.4.2](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-61 April release)
Ubuntu 16	f8f616147c9daaaeb9cd8cb1ae44157f
Ubuntu 18	1de5b0baf4edec8d7817bddcecf3824
Ubuntu 20	829e91c7fac1a0ab7e36cb4a65d19906
RedHat Enterprise Linux (RHEL) 7	477f1d3161ea16e8d52c41eb61d92637
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	1170af43f88bf5f9095890916c0fe74d
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	
Amazon Linux 2 (Release Candidate)	e246ae158db32e0a483a23392b2a7f47

New features and enhancements

- Amazon Linux 2 supported as a release candidate (RC)



Note: A database with modules cannot reside on an Amazon Linux 2 (release candidate) node. Support will be added in a future maintenance release.

- Add node ID indication to [debug_info package](#) (RS95360)
- Add support for underscore _ as a valid character for [rack awareness](#) (RS87458)
- When updating a [BDB object](#), the version property is immutable. Any validation will be performed according to the BDB object version that was set during upgrade or install (RS93294)

Redis Stack v6.2.6

Redis Enterprise Software v6.4.2 includes the new features delivered in the latest [Redis Stack release \(6.2.6 v6\)](#):

- [RediSearch v2.6.6](#)
- [RedisJSON v2.4.6](#)
- [RedisBloom v2.4.5](#)
- [RedisGraph v2.10.9](#)
- [RedisTimeSeries v1.8.9](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Version changes

Deprecations

Ubuntu 16.04

Ubuntu 16 support is deprecated and will be removed in a future release.

Active-Active database persistence

The RDB snapshot option for [Active-Active database persistence](#) is deprecated and will be removed in a future release.

Please plan to reconfigure any Active-Active databases to use append-only file (AOF) persistence with the following command:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> \
--default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

Resolved issues

- RS95344 - Fixed known limitation for RHEL 7 and RHEL 8 where CRDB databases fail to start when installed using custom installation paths
- RS88010 - Roll back node configuration when the [remove node](#) operation fails
- RS95824 - Fixed an issue with running `rl_rdbconvert` on Ubuntu
- RS97971 - Fixed known limitation for the edge case where resharding fails for rack-aware databases with no replication

Known limitations

Operating system limitations

Ubuntu 20.04

By default, you cannot use the SHA1 hash algorithm ([OpenSSL's default security level is set to 2](#)). The operating system will reject SHA1 certificates even if the `mtls_allow_weak_hashing` option is enabled. You need to replace SHA1 certificates with newer certificates that use SHA-256. Note that the certificates provided with Redis Enterprise Software use SHA-256.

Modules not supported for Amazon Linux 2 release candidate

A database with modules cannot reside on an Amazon Linux 2 (release candidate) node. Support will be added in a future maintenance release.

Updated: August 17, 2023

Redis Enterprise Software release notes 6.4.2-43 (March 2023)

This is a maintenance release for [Redis Enterprise Software version 6.4.2](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-43 March release)
Ubuntu 16	bb3d3ab7590834790d2d9d9e3dcdf788
Ubuntu 18	55e99e6db76277c15fa46c49d58995b5
Ubuntu 20	ccfa62ca7ea955b8c19d9c40bdc1fea
RedHat Enterprise Linux (RHEL) 7	eed4083b0b4c066d31ea9b8ba60a34f5
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	7fd4321f6a5502c76197fc039f44ffa3
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	

New features and enhancements

Support for Ubuntu 20

Added operating system support for Ubuntu 20.04.

Validate client certificates by subject attributes

- Added the ability to enter multiple `Organizational Unit` (OU) values to a certificate validation's subject entry. Subject validation only succeeds if all OU values match.
- Validation fails if the certificate's subject contains an extra field which is missing from the certificate's configured subject (empty field).

Safe node removal (RS92095)

As of this release, the node removal task waits for persistence files to be created on a new node before finishing removal. For more information, see [Remove a cluster node](#) and `radmin node remove`.

Allow port configuration for `gossip_envoy` service (RS72028)

You can use the REST API to configure the `gossip_envoy_admin_port` during bootstrapping or runtime. See the [cluster object reference](#) for more details.

Support for `radmin` will be added in a future 6.4.2 maintenance release.

Redis Stack v6.2.6

Redis Enterprise Software v6.4.2 includes the new features delivered in the latest [Redis Stack release \(v6.2.6\)](#):

- [RediSearch v2.6.4](#)
- [RedisJSON v2.4.5](#)
- [RedisBloom v2.4.3](#)
- [RedisGraph v2.10.5](#)
- [RedisTimeSeries v1.8.5](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Resolved issues

- RS95344 - Fix custom installation linkage bug for files related to Active-Active
- RS91554 - Enhance visibility of `redis_mgr`
- RS90496 - Improve cross-node communication to support failovers
- RS93004 - Allow positive `audit_port` configuration
- RS58918 - Reduce the number of clients blocked by flushing a database. Only the client that requested the flush will be blocked until the action finishes. Other clients won't be affected.
- RS85447 - When creating a database, the memory size is total memory for the database, not the total dataset size. Therefore, if you enable replication and the database has two shards, each shard will have half of the allocated memory size.
- RS90556 - Heartbeats - upgrade `sysinfo` dependency 0.19.1 -> 0.27.7
- RS62552 - Fix database authentication failures for LDAP users when the password contains the% character
- RS93566 - Added `rlcheck` validation step to verify that all needed modules (modules used during the initial creation of any CRDBs on the cluster) are actually available on the cluster. In some cases, if a specific module version is required, the user will need to upload it in order to upgrade the cluster. After the upgrade is complete, you can delete the required module.

Known limitations

Feature limitations

Resharding fails for rack-aware databases with no replication

When a database is configured as `rack-aware` and replication is turned off, the resharding operation fails.

RS97971 - This limitation will be fixed in a future 6.4.2 maintenance release.

Workaround:

Before resharding your database, turn off rack awareness:

```
curl -k -u "<user>:<password>" -H "Content-type: application/json" -d '{"rack_aware": false}' -X PUT "https://localhost:9443/v1/b dbs/<bdb_uid>"
```

After the resharding process is complete, you can re-enable rack awareness:

```
curl -k -u "<user>:<password>" -H "Content-type: application/json" -d '{"rack_aware": true}' -X PUT "https://localhost:9443/v1/b dbs/<bdb_uid>"
```

Operating system limitations

RHEL 7 and RHEL 8

RS95344 - CRDB database will not start on Redis Enterprise v6.4.2 with a custom installation path.

For a workaround, use the following commands to add the relevant CRDB files to the Redis library:

```
$ yum install -y chrpath  
$ find $installdir -name "crdt.so" | xargs -n1 -I {} /bin/bash -c 'chrpath -r ${libdir} {}'
```

This limitation will be fixed in a future 6.4.2 maintenance release.

Ubuntu 20.04

By default, you cannot use the SHA1 hash algorithm ([OpenSSL's default security level is set to 2](#)). The operating system will reject SHA1 certificates even if the `mtls_allow_weak_hashing` option is enabled. You need to replace SHA1 certificates with newer certificates that use SHA-256. Note that the certificates provided with Redis Enterprise Software use SHA-256.

Ubuntu 16, 18, 20

RS95824 - An error occurs when running `rl_rdbconvert` manually.

As a workaround, run the following command:

```
LD_LIBRARY_PATH=$libdir rl_rdbconvert
```

Updated: August 17, 2023

Redis Enterprise Software release notes 6.4.2-30 (February 2023)

Redis Enterprise Software version 6.4.2 is now available!

This version offers:

- Extended validation of client certificates via mTLS (mutual TLS) full subject support
- Support for default restrictive permissions when using publish/subscribe commands and ACLs (access control lists)
- Enhanced TLS performance when Redis returns large arrays in responses
- Compatibility with [open source Redis 6.2.7](#)
- Additional enhancements and bug fixes

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.4.2-30 February release)
Ubuntu 16	b0dbecaa974ca08245dda55d53b6fe9b
Ubuntu 18	a5192e8b0734db80d6b7c2b98a170c58

Package	MD5 checksum (6.4.2-30 February release)
RedHat Enterprise Linux (RHEL) 7 Oracle Enterprise Linux (OL) 7	c1537855dcfe7a7cedf9031ce01e2b9b
RedHat Enterprise Linux (RHEL) 8 Oracle Enterprise Linux (OL) 8 Rocky Enterprise Linux	a24dc749d6dcb5df2162d7a41791c7aa

New features and enhancements

Validate client certificates by subject attributes

You can now validate client certificates by their Subject attributes. When a client attempts to connect to a database, Redis Enterprise Software compares the values of the client certificate subject attributes to the subject values allowed by the database. Clients can connect to the database only if the subject values match. This gives more flexibility in controlling which clients can access which databases.

See [Enable TLS](#) for more information.

Default pub/sub ACL permissions

Redis is continuously enhancing its ACL (access control list) functionality and coverage. Redis version 6.2 enhances [ACLs](#) to allow and disallow pub/sub channels.

Part of protecting pub/sub channels requires changing the default access from permissive to restrictive, which blocks all pub/sub channels unless specifically permitted by an ACL rule. To allow this transition across all databases in the cluster, Redis Enterprise Software 6.4.2 provides a new configuration option `acl-pubsub-default` that sets the cluster-wide default for all channels to either permitted or restricted.

The 6.4.2 installation-provided value of `acl-pubsub-default` is permissive (`allchannels`) to comply with earlier Redis versions. After you upgrade all databases in the cluster to Redis DB version 6.2 (or later in future versions), you can use `radmin` or the REST API to change the value to restrictive (`resetchannels`).

To allow certain users to access specific pub/sub channels, define the appropriate ACL. Redis Enterprise Software 6.4.2 enhances the admin console (UI), CLI, and REST API to support pub/sub channel ACL definitions.

If you use ACLs and pub/sub channels, we recommend you review your databases and ACL settings and plan to change your cluster to restricted mode. This will help you prepare for future Redis Enterprise Software releases that use restrictive `resetchannels` as the new default for `acl-pubsub-default`.

Redis modules

Redis Enterprise Software v6.4.2 includes the following Redis modules:

- [RedisSearch v2.4.16](#)
- [RedisJSON v2.2.0](#)
- [RedisBloom v2.2.18](#)
- [RedisGraph v2.8.20](#)
- [RedisTimeSeries v1.6.17](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Installations, upgrades, and troubleshooting

- Added the ability for `install.sh` to run even if “upgrade mode” is already enabled to allow reruns in case of a previous run failure (RS77319)
- Added log messages to the `redis_mgr` process (RS77891), the `job_scheduler` process (RS82673), and the `install.sh` script (RS82673)
- Improved `radmin` error messages for certificate validation (RS79933)
- Added internode encryption ports to command-line utility `rlcheck` validation (RS68965)
- Added an alert to notify when a node operation (such as maintenance mode) failed, aborted, or was canceled. The alert is enabled by default (RS76089)

Version changes

Breaking changes

- REST API: the `authorized_names` field of the [BDB object](#) is deprecated. Use the new `authorized_subjects` field instead.

New default Redis DB version

Both Redis Enterprise Software versions 6.2.x and 6.4.x package two Redis DB versions: Redis DB 6.0 and Redis DB 6.2. Until now, the default Redis DB version for creating new databases and upgrading existing databases was 6.0 (enforced by the `redis_upgrade_policy` parameter).

To allow customers more flexibility in future upgrades, starting with Redis Enterprise Software 6.4.2, the default Redis DB version for new and upgraded databases is now 6.2 for all upgrade policies (`redis_upgrade_policy=major` and `redis_upgrade_policy=latest`).

Redis Enterprise	Bundled Redis DB versions	Default DB version (upgraded/new databases)
6.2.x	6.0, 6.2	6.0
6.4.2	6.0, 6.2	6.2

You can override the default version with [rladmin](#); however, we recommend that you don't change this setting.

Deprecations

Ubuntu 16.04

Ubuntu 16 support is considered deprecated and will be removed in a future release. Ubuntu 16.04 LTS (Xenial) has reached the end of its free initial five-year security maintenance period as of April 30, 2021.

Active-Active database persistence

The snapshot option for Active-Active database persistence is deprecated. We advise customers running Active-Active databases, configured with snapshot data persistence, to reconfigure their data persistence mode to use the AOF (Append Only File) option with the following command:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> \
--default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later. Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES. Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release. Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

Resolved issues

- RS72866 - Improved performance for client connections which use TLS
- RS78241 - Fixed shard placement to always respect rack-zone restrictions and avoid a state where a primary (master) and replica are on the same rack, even if temporarily
- RS78144 - Removed the dependency on system-wide `ldconfig` so non-interactive processes will use their own dynamic libraries without impacting external services
- RS78028 - Fixed race condition during rolling upgrade that might result in shards repeatedly restarting
- RS77964 - Fixed module deletion to remove the old directory with the module
- RS75259 - Fixed node to prevent using plain text communication instead of TLS after losing connectivity
- RS69616 - Fixed validation for internode communication ports
- RS83535 - Fixed `sentinel_service` to start on RHEL 8 with DISA STIG profile
- RS87191 - Fixed a cross slot error when using Auto Tiering with Replica Of, in case a key on the source database swapped from RAM to flash and expired while it was also part of Lua script execution

Known limitations

Feature limitations

- RS101204 - High memory consumption caused by the `persistence_mgr` service when AOF persistence is configured for every second. Monitor RAM usage of the process. In case of high usage, the temporary workaround is to restart the service by running `supervisorctl restart persistence_mgr`. A permanent fix is to install or upgrade to the 6.4.2 June maintenance release.

Upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.4.2-30, perform the following steps:

1. Use `crdb-cli crdb get --crdb-guid <crdb-guid>`

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.

3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

This limitation has been fixed and resolved as of [v6.4.2-43](#).

Operating system limitations

RHEL 7 and RHEL 8

If you have a custom installation with a non-default `$installdir` and use Active-Active or Auto Tiering features, failures might occur when you upgrade. This issue will be fixed in a future maintenance release.

RHEL 8

Due to module binary differences between RHEL 7 and RHEL 8, you cannot upgrade RHEL 7 clusters to RHEL 8 when they host databases using modules. Instead, you need to create a new cluster on RHEL 8 and then migrate existing data from your RHEL 7 cluster. This does not apply to clusters that do not use modules.

Security

Open source Redis security fixes compatibility

As part of Redis's commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with [open source Redis](#). The following open source Redis [CVEs](#) do not affect Redis Enterprise:

- [CVE-2021-32625](#) – Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) – Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32675](#) – Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) – Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) – Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the `proto-max-bulk-len` CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on the [Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

Redis Enterprise has already included the fixes for the relevant CVEs. Some CVEs announced for open source Redis do not affect Redis Enterprise due to different and additional functionality available in Redis Enterprise that is not available in open source Redis.

Redis Enterprise Software release notes 6.2.18

Redis Enterprise Software version 6.2.18 is now available!

This version of Redis Enterprise Software offers:

- RedisJSON on Active-Active General Availability
- Database connection auditing
- Private key encryption
- Active-Active support for `memory usage` command
- `crdb-cli` improvements
- Compatibility with open source Redis v6.2.6
- Additional enhancements and fixes

Detailed release notes

For more detailed release notes, select a build version from the following table:

Version (Release date)	Major changes	OSS Redis compatibility
6.2.18-70 (January 2023)	New script to generate new self-signed certificates for renewal. Improved logs. Deactivate <code>alert_mgr</code> using <code>rladmin</code> .	Redis 6.2.6
6.2.18-65 (December 2022)	Added <code>To</code> field to test email server alerts. New flag to skip cluster resource validation during imports. Improved error messages.	Redis 6.2.6
6.2.18-58 (November 2022)	UI support for custom REST API port. Added info level for troubleshooting <code>redis_mngr</code> .	Redis 6.2.6
6.2.18-49 (October 2022)	New REST API endpoint to check whether a node is a primary or a replica. Added metrics to track certificate expiration time.	Redis 6.2.6
6.2.18-43 (September 2022)	Database auditing. Private key encryption. Active-Active database support for <code>MEMORY USAGE</code> command. Improvements to <code>crdb-cli</code> .	Redis 6.2.6

Deprecations

Active-Active database persistence

The snapshot option for [data persistence on Active-Active databases](#) will be deprecated in a future version of Redis Enterprise Software. If you have an Active-Active database using snapshot persistence, switch to AOF persistence. Use `crdb-cli` to do so:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> --default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later.

Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release.

Certain operating systems, such as RHEL 8, have removed support for the earlier protocols. Redis Enterprise Software does not support connection protocols that are not supported by the underlying operating system.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES.

Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release.

Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

Known limitations

Feature limitations

- RS54131 Running the QUIT command on a TLS connected database closes connection and does not return a +OK reply
- An intermittent issue can occur where a CRDB process becomes stuck in a restart loop. If this issue occurs while upgrading to Redis Enterprise Software version 6.2.18, please upgrade to the latest version [6.4.2-69](#) or [contact support](#).
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use `radmin cluster config` to configure handle_redirects or handle_metrics_redirects.
- RS51144, RS102128 - Active-Active: To start successfully, the syncer (`crdt-syncer`) must connect to all sources. In multi-cluster configurations (more than 2 A-A clusters participating), in some cases, if one or more of the clusters is not available, A-A replication will be down.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use `crdb-cli` to verify that the modules (modules) and their versions (in module_list) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

Updated: August 17, 2023

Redis Enterprise Software release notes 6.2.18-70 (January 2023)

This is a maintenance release for [Redis Enterprise Software version 6.2.18](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.2.18-70 January release)
Ubuntu 16	69d2d2c71232adb15cebf29308ac54da
Ubuntu 18	22e0637107a32ccb96a704abe9650adf
RedHat Enterprise Linux (RHEL) 7	e14fcf6973418602f2b64a55a0bc8374
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	f61d0d8f0bb5ad90a470482d6575eb27
Oracle Enterprise Linux (OL) 8	
Rocky Enterprise Linux	

New features and enhancements

- Added a new script to generate new self-signed certificates for renewal. For details, see [Certificates](#)
- Added rpm -q command output to the logs (RS82673)
- Added logs for server operations: start, stop, restart (RS87775)
- Added alert_mgr to the services you can deactivate with `radmin` (RS89572)

Resolved issues

- RS61062 - Remove leading colon added to PATH variable

Known upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use `crdb-cli crdb get --crdb-guid <crdb-guid>` to verify that the modules (modules) and their versions (in `module_list`) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

Updated: April 10, 2023

Redis Enterprise Software release notes 6.2.18-65 (December 2022)

This is a maintenance release for [Redis Enterprise Software version 6.2.18](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.2.18-65 December release)
Ubuntu 16	515c7bbc97eaf1939757c8894eda9523
Ubuntu 18	830e8704c0f6902a04df9ff53cc5e41f
RedHat Enterprise Linux (RHEL) 7	b51d13e30f4a7e1a134982342b8865b0
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	762f907c1f0e6e0e88e4ae5bf967e143
Oracle Enterprise Linux (OL) 8	

New features and enhancements

- When setting up an alert email server in the UI: added a To field to the `test_email_server` function (RS86119)
- Added a new flag to skip cluster resource validation when performing import (RS88086)
- Enhanced error message information for connection handling
- Added logging details to `insufficient_resources` planner error

Redis modules

Redis Enterprise Software v6.2.18-65 (December release) includes newer versions of the following Redis modules:

- [RediSearch v2.4.16](#)
- [RedisGraph v2.8.20](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Resolved issues

- RS87191 - Fixes syncer stop due to cross slot violation error in Auto Tiering

Known upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use crdb-cli to verify that the modules (modules) and their versions (in module_list) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help,[contact Redis support](#) before taking additional steps.

Updated: August 17, 2023

Redis Enterprise Software release notes 6.2.18-58 (November 2022)

This is a maintenance release for [Redis Enterprise Software version 6.2.18](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.2.18-58 November release)
Ubuntu 16	38d974c47004df0808b71c2bd5a96996
Ubuntu 18	6a5ca7a287f1808a50d13312162aac4b
RedHat Enterprise Linux (RHEL) 7	d8283a3475a70823641249e3cfe3ab6e
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	040e45c866fde391b35ae85d648a4afa
Oracle Enterprise Linux (OL) 8	

New features and enhancements

- Added support to the UI to work with custom REST API port (RS84428)
- Added info level print to redis_mgr for troubleshooting (RS85385)

Redis modules

Redis Enterprise Software v6.2.18-58 (November release) includes newer versions of the following Redis modules:

- [RediSearch v2.4.14](#)
- [RedisGraph v2.8.19](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Resolved issues

- RS85369 - Fixes umask validation during installation to allow for temporary umask change
- RS77339 - Fixes the "FROM" email addresses of email alerts
- RS86005 - Fixes bdb</int:uid>/actions/recover API to return the bdb status and not the recovery_plan

Known upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use crdb-cli to verify that the modules (modules) and their versions (in module_list) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help,[contact Redis support](#) before taking additional steps.

Redis Enterprise Software release notes 6.2.18-49 (October 2022)

This is a maintenance release for [Redis Enterprise Software version 6.2.18](#).

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.2.18-49 October release)
Ubuntu 16	89029a28a2b0e3ad379559b94d00ae32
Ubuntu 18	830e8704c0f6902a04df9ff53cc5e41f
RedHat Enterprise Linux (RHEL) 7	766d2f8448fc28d0ea67cac20387a9e3
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	7e31fb3e3416b404513dad508669f836
Oracle Enterprise Linux (OL) 8	

New features and enhancements

- The [/nodes/status](#) REST API endpoint indicates whether a node is the primary. (RS82566)
- Added metrics to track certificates expiration time (RS56040)

Redis modules

Redis Enterprise Software v6.2.18-49 (October release) includes the following Redis modules:

- [RedisSearch v2.4.11](#)
- [RedisJSON v2.2.0](#)
- [RedisBloom v2.2.18](#)
- [RedisGraph v2.8.17](#)
- [RedisTimeSeries v1.6.17](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Known upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use `crdb-cli` to verify that the modules (`modules`) and their versions (in `module_list`) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's `redis modules` tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

Updated: April 10, 2023

Redis Enterprise Software release notes 6.2.18-43 (September 2022)

[Redis Enterprise Software version 6.2.18](#) is now available!

This version of Redis Enterprise Software offers:

- RedisJSON on Active-Active General Availability
- Database connection auditing
- Private key encryption
- Active-Active support for `memory usage` command
- `crdb-cli` improvements
- Compatibility with open source Redis v6.2.6
- Additional enhancements and fixes

The following table shows the MD5 checksums for the available packages:

Package	MD5 checksum (6.2.18-43 September release)
Ubuntu 16	055973eb7009073b0c199ec1dfd81018
Ubuntu 18	8c37c6ae10b0ae4956e3c11db80d18ce
RedHat Enterprise Linux (RHEL) 7	c770a66d9bfdd8734f1208d64aa67784
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	85eb6339f837205d83f215e32c1d028f
Oracle Enterprise Linux (OL) 8	

New features and enhancements

General Availability of Active-Active databases with RedisJSON

Active-Active databases now support Index, query, and full-text search of nested JSON documents when combining RedisJSON with RediSearch.

Database connection auditing

You can now [audit database connection](#) and authentication events to track and troubleshoot activity. Administrators can now use third-party systems to track and analyze connection events in realtime.

Private key encryption

When enabled, private key encryption encrypts private keys stored in the cluster configuration store (CCS). Private keys are encrypted using a secure, self-contained internal process. Databases must be at least version 6.2.2 or later to use this feature.

Active-Active support for MEMORY USAGE command

Redis Enterprise Active-Active databases now support the [MEMORY USAGE](#) command, which simplifies troubleshooting and lets applications detect anomalous behavior and dangerous trends. To learn more, see [Active-Active with RedisJSON](#).

MEMORY USAGE reports the RAM memory use, in bytes, of a key and its value. The result includes the memory allocated for the data value and the administrative overhead associated with the key.

crdb-cli improvements

The `crdb-cli` utility used to manage Active-Active databases now allows greater visibility into management operations to make it easier to investigate and troubleshoot problems. You can now use:

- `crdb-cli task list` to retrieve details about current and previous tasks for all Active-Active databases on a cluster
- `crdb-cli task --task-id <task-id>` to get detailed information about a specific task.

To learn more, see [crdb-cli](#)

Redis modules

Redis Enterprise Software v6.2.18-43 (September release) includes the following Redis modules:

- [RediSearch v2.4.11](#)
- [RedisJSON v2.2.0](#)
- [RedisBloom v2.2.18](#)
- [RedisGraph v2.8.17](#)
- [RedisTimeSeries v1.6.17](#)

See [Upgrade modules](#) to learn how to upgrade a module for a database.

Additional enhancements

- Added support for the MODULE LIST command on Active-Active databases
- Enhanced validity checks of the input parameters of the CRDB-CLI tool
- The Syncer lag calculation has been improved and fixes multiple miscalculations
- Support package includes additional information for Active-Active databases
- Added the ability to retrieve the syncer mode (centralized / distributed) by running

```
rladmin info db [{db:<id> | <name>}]
```

To learn more, see [Distributed synchronization](#)

Version changes

Breaking changes

RS84006 - When using the REST API to create a database, this specific set of conditions can lead to an error:

- sharding is enabled
- oss_sharding and implicit_shard_key are both deactivated
- A shard_key_regex is not defined

If this occurs, the database endpoint returns (error) ERR key "test" does not match any rule.

To address this issue, do one of the following:

- Explicitly define shard_key_regex
- Enable oss_sharding or implicit_shard_key (enabling both also works)

Prerequisites and notes

- You can [upgrade to v6.2.18](#) from Redis Enterprise Software v6.0 and later.
- Refer to [v6.2.4 release notes](#) for important notes regarding changes made to the upgrade policy and how those changes might impact your experience.
- Upgrades from versions earlier than v6.0 are not supported.
- If you plan to upgrade your cluster to RHEL 8, see the [v6.2.8 release notes](#) for known limitations.

Deprecations

Active-Active database persistence

The snapshot option for [data persistence on Active-Active databases](#) will be deprecated in a future version of Redis Enterprise Software. If you have an Active-Active database using snapshot persistence, we strongly encourage you to switch to AOF persistence. Use crdb-cli to do so:

```
crdb-cli crdb update --crdb-guid <CRDB_GUID> --default-db-config '{"data_persistence": "aof", "aof_policy": "appendfsync-every-sec"}'
```

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later.

Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release.

Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

3DES encryption cipher

The 3DES encryption cipher is considered deprecated in favor of stronger ciphers like AES.

Please verify that all clients, apps, and connections support the AES cipher. Support for 3DES will be removed in a future release.

Certain operating systems, such as RHEL 8, have already removed support for 3DES. Redis Enterprise Software cannot support cipher suites that are not supported by the underlying operating system.

Product lifecycle updates

Redis Enterprise Software v6.0.x reached end-of-life (EOL) on May 31, 2022.

EOL of Redis Enterprise Software 6.2.x was reset to occur 18 months after the release of version 6.4, in accordance with the updated EOL policy. The new EOL date is August 31, 2024.

To learn more, see the Redis Enterprise Software [product lifecycle](#), which details the release number and the end-of-life schedule for Redis Enterprise Software.

For Redis modules information and lifecycle, see [Module lifecycle](#).

Resolved issues

- RS64002 - Fixes email alerts for LDAP mappings.
- RS79519 - fixes a bug that prevented Administrators from editing an Active-Active database using the API with username instead of email as their identifier.
- RS78039 - fixes a bug that caused the sync between two shards to pause when resetting a shard's data which can be when performing full sync or importing an RDB.
- RS73454 - Updates internal timeouts to enable faster resharding.
- RS75783 - Fixes failover due to false identification of dead nodes when master node goes down.
- RS75206, RS52686 - Fixes backup_interval_offset in case where the user chose an offset that is higher than backup_interval; Fixes the UI from resetting backup_interval_offset after manual DB configuration.
- RS75176 - Fixes rare case of stuck state machine during "maintenance off".
- RS57200 - Add an IP address to "failed_authentication_attempt" errors.
- RS56615 - Changed radmin tune db db_name max_aof_load_time to receive the value in seconds; Added max_aof_load_time option to radmin help tune.
- RS54745 - Fixes the Rest API to reject BDB creation using negative integers as a uid.
- RS46092 - Fixes rlcheck failure when somaxconn policy is a value other than 1024.
- RS68965, RS80615 - Adds internode encryption ports 3340-3344 to rlcheck connectivity.
- RS63302 - Adds umask validation for root user when installing.
- RS46947 - Fixes removal of old installations in install.sh.

Known limitations

Feature limitations

- RS81463 A shard might crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.
- RS54131 Running the QUIT command on a TLS connected database closes connection and does not return a +OK reply

Upgrade limitations

Before you upgrade a cluster that hosts Active-Active databases with modules to v6.2.18, perform the following steps:

1. Use crdb-cli to verify that the modules (modules) and their versions (in module_list) are as they appear in the database configuration and in the default database configuration:

```
crdb-cli crdb get --crdb-guid <crdb-guid>
```

2. From the admin console's **redis modules** tab, validate that these modules with their specific versions are loaded to the cluster.
3. If one or more of the modules/versions are missing or if you need help, [contact Redis support](#) before taking additional steps.

Security

Open source Redis security fixes compatibility

As part of Redis's commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with open source Redis. The following [Open Source Redis CVEs](#) do not affect Redis Enterprise:

- [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

- [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16) security fixes for [recent CVEs](#). Redis Enterprise has already included the fixes for the relevant CVEs. Some CVEs announced for Open Source Redis do not affect Redis Enterprise due to different and additional functionality available in Redis Enterprise that is not available in Open Source Redis.

Updated: August 17, 2023

Redis Enterprise Software release notes 6.2.12 (August 2022)

Redis Enterprise Software version 6.2.12 is now available!

This version of Redis Enterprise Software offers:

- OCSP stapling of the server proxy certificate
- Password and session configuration settings via the admin console
- Compatibility with open source Redis v6.2.6
- Support for Red Hat Enterprise Linux (RHEL) v8.6
- Additional enhancements and fixes

The following table shows the MD5 checksums for the available packages.

Package	MD5 Checksum
Ubuntu 16	e702c906f200940e06ef031e6b8006d9
Ubuntu 18	7ea70067e8828b59336380df087fe03d
RedHat Enterprise Linux (RHEL) 7	8ffda6186f70354b9d10c1ce43938c3c
Oracle Enterprise Linux (OL) 7	
RedHat Enterprise Linux (RHEL) 8	334fe7979a7376b28fcf48913403bfb7
Oracle Enterprise Linux (OL) 8	

Features and enhancements

- Server side OCSP Stapling

[Online Certificate Status Protocol](#) (OCSP) helps verify the status of certificates managed by a third-party [certificate authority](#) (CA). It tells you whether certificates are valid, revoked, or unknown.

Redis Enterprise Software v6.2.12 implements OCSP stapling, which allows clients to validate the status of a server proxy certificate. When OCSP is enabled, the Redis Enterprise server regularly polls the CA OCSP responder to determine a certificate's status. The response is cached until the next polling attempt; the cached value is served to clients during the [TLS handshake](#).

To learn more, see [Enable OCSP stapling](#).

- Session and security attributes in the admin console

You can now use the admin console to configure [password complexity rules](#), [user login lockout](#), and [session timeout](#).

- Mount point import enhancement

When importing data, Redis Enterprise copies files to a temporary directory on the node. For mount point import sources only, Redis Enterprise now reads files directly from the mount point. Because this import method does not copy files to a temporary directory, nodes do not require extra disk space. This new behavior is enabled by default and does not require configuration.

Version changes

Prerequisites and notes

- You can [upgrade to v6.2.12](#) from Redis Enterprise Software v6.0 and later.
- Refer to [v6.2.4 release notes](#) for important notes regarding changes made to the upgrade policy and how those changes might impact your experience.
- Upgrades from versions earlier than v6.0 are not supported.
- If you are using the earlier cluster-based LDAP mechanism, you *must* migrate to the role-based mechanism *before* upgrading to v6.2.12. For details, see [Migrate to role-based LDAP](#).
- If you plan to upgrade your cluster to RHEL 8, see the [v6.2.8 release notes](#) for known limitations.

Future deprecation notice

TLS 1.0 and TLS 1.1

TLS 1.0 and TLS 1.1 connections are considered deprecated in favor of TLS 1.2 or later.

Please verify that all clients, apps, and connections support TLS 1.2. Support for the earlier protocols will be removed in a future release.

Certain operating systems, such as RHEL 8, have already removed support for the earlier protocols. Redis Enterprise Software cannot support connection protocols that are not supported by the underlying operating system.

Product lifecycle updates

Redis Enterprise Software v6.0.x will reach end of life (EOL) on May 31, 2022.

To learn more, see the Redis Enterprise Software [product lifecycle](#), which details the release number and the end-of-life schedule for Redis Enterprise Software.

For Redis modules information and lifecycle, see [Module lifecycle](#).

Redis modules

Redis Enterprise Software v6.2.12 includes the following Redis modules:

- [RediSearch v2.4.9](#)
- [RedisJSON v2.0.11](#)
- [RedisBloom v2.2.17](#)
- [RedisGraph v2.8.15](#)
- [RedisTimeSeries v1.6.16](#)

For help upgrading a module, see [Add a module to a cluster](#).

Interface enhancements

- Allow creation and editing of sharded databases with a single shard. This lets you prepare for future scaling by ensuring your apps work in clustering mode, regardless of the actual number of configured shards
- Enhanced the slowlog in the UI to display unprintable characters in RESTORE commands
- Added support for custom paths when bootstrapping the cluster via the UI
- Improve readability of geo-distributed log entries in the UI

Additional enhancements

- Added support for the MODULE LIST command on Active-Active databases
- Enhanced validity checks of the input parameters of the CRDB-CLI tool

Resolved issues

- RS38320 A failed task leaves the nodes list outdated in the UI
- RS73768, RS72082 Increased certificate rotation timeout to allow it to finish
- RS72466, RS68668 Fix false positive alerts for certification expiration
- RS69256 Change pre-bootstrap default TLS version to 1.2
- RS67133 Fixed replication for command RESTOREMODAUX
- RS66468 Fixed “Test regex keys” in the UI
- RS58156 Fixed the installation to abort and alert when encountering issues in NTP
- RS67935 Fixed releasing 30MB of memory when deleting an Active-Active database
- RS64276 Fixed high memory consumption of the DMC output buffers when running CLIENT LIST

Known limitations

- RS81463 A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.
- RS54131 Returning +OK reply from the QUIT command on a TLS enabled database
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use `radmin cluster config` to configure handle_redirects or handle_metrics_redirects.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory /etc/opt/redislabs/ even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Security

Open Source Redis Security fixes compatibility

As part of Redis commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with open source Redis. The following [Open Source Redis CVEs](#) do not affect Redis Enterprise:

- [CVE-2022-24834](#) - Redis Enterprise versions 6.2.12 and later are not impacted by the CVE that was found and fixed in open source Redis. A fix to prevent Lua scripts from causing heap overflow was implemented in Redis Enterprise version 6.2.12. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 7.0.12, Redis 6.2.13, Redis 6.0.20)
- [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16) security fixes for [recent CVEs](#).

Redis Enterprise has already included the fixes for the relevant CVEs. Some CVEs announced for open source Redis do not affect Redis Enterprise due to different and additional functionality available in Redis Enterprise that is not available in open source Redis.

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.2.10 (February 2022)

Redis Enterprise Software version 6.2.10 is now available!

The following table shows the MD5 checksums for the available packages.

Package	MD5 Checksum
Ubuntu 16	531cea69a58fbc1125bc5f76ba01da7f
Ubuntu 18	ec9ac6e0111dc85605d3b98e83f50150

Package	MD5 Checksum
RedHat Enterprise Linux (RHEL) 7 Oracle Enterprise Linux (OL) 7	2f7572caab9600417ef8b4ee474d6768
RedHat Enterprise Linux (RHEL) 8 Oracle Enterprise Linux (OL) 8	377a539ee050515e1e0640dec1e04129
K8s Ubuntu	099192416a70a12790535bcdcd78a6e87
K8s RHEL	f267abe81770ddf36f022232f4c2cb2e

Features and enhancements

- Upgrade the Redis Enterprise infrastructure to [Python v3.9](#).
- [Red Hat Enterprise Linux \(RHEL\) v8.5](#) and [Red Hat Enterprise Linux \(RHEL\) v8.6](#) is now a supported platform.
- [Oracle Linux v8](#) is now a supported platform.
- Compatibility with [open source Redis 6.2.5](#).
- Compatibility with the [security fixes](#) of the latest [open source Redis 6.2.6](#).
- Enhancements and bug fixes.

Version changes

Prerequisites and notes

- You can [upgrade to v6.2.10](#) from Redis Enterprise Software v6.0 and later.
- Refer to [v6.2.4 release notes](#) for important notes regarding changes made to the upgrade.
- Upgrades from versions earlier than v6.0 are not supported.
- If you plan to upgrade your cluster to RHEL 8, refer to [v6.2.8 release notes](#) for known limitations.
- If you are using Active-Active or Active-Passive (ReplicaOf) databases and experience synchronization issues as a result of the upgrade, see RS67434 details in [Resolved issues](#) for help resolving the problem.

Product lifecycle updates

Redis Enterprise Software v6.0.x will reach end of life (EOF) on May 31, 2022.

To learn more, see the Redis Enterprise Software [product lifecycle](#), which details the release number and the end-of-life schedule for Redis Enterprise Software.

For Redis modules information and lifecycle, see [Module lifecycle](#).

Redis modules

Redis Enterprise Software v6.2.10 includes the following Redis modules:

- [RediSearch v2.2.6](#)
- [RedisJSON v2.0.6](#)
- [RedisBloom v2.2.9](#)
- [RedisGraph v2.4.12](#)
- [RedisTimeSeries v1.4.13](#)

Starting with Redis Enterprise Software v6.2.10 build 121, the included modules versions are:

- [RediSearch v2.4.6](#)
- [RedisJSON v2.0.8](#)
- [RedisBloom v2.2.14](#)
- [RedisGraph v2.8.12](#)
- [RedisTimeSeries v1.6.9](#)

For help upgrading a module, see [Add a module to a cluster](#).

Interface enhancements

- When choosing RedisJSON, the user interface (UI) now suggests RedisSearch as well. To learn more, see the [RedisJSON preview announcement](#), which details the benefits of combining [RedisJSON](#) and [RedisSearch](#).
- Adds the ability to sort the columns of the node list (RS48256).
- When creating a new geo-distributed (Active-Active) database, an endpoint port is no longer required. The system assigns one if none is provided (RS27632).

Additional enhancements

- Added an option to run a connectivity health check for the management layer of Active-Active databases. Run the following REST API command:

```
GET https://[host][:port]/v1/crdb/<crdb_guid>/health_report
```

- Added TLS handshake error messages to the DMC proxy log (RS59346).

Resolved issues

- RS58219 - Fixes a UI error message that showed a path instead of a relevant error message.
- RS44958 - Fixes incorrect description for the graph “incoming traffic” in Active-Active (geo-distributed) database UI Metrics.
- RS66280 - Fixes the lexicographic [SORT](#) command on Active-Active databases (e.g. `SORT mylist ALPHA`). The `SORT` command should only run on keys mapped to the same slot.
- RS64575 - Fixes a bug in the replication between primary and replica shards of a destination Active-active database in the scenario of using Replica-Of from a single to an Active-Active database, where the syncer process went down during the full sync.
- RS65370 - Adds logic to remove old syncer entries in the cluster configuration during upgrades.
- RS67434 - Version 6.2.10 fixes the mTLS handshake between the [syncer process](#) and the [proxy \(DMC\)](#), where the proxy presented a leaf certificate without its full chain to the syncer. After upgrading to 6.2.10, syncer connections using invalid certificates will break the synchronization between Active-Active instances or deployments using Replica Of when TLS is enabled. To ensure certificates are valid before upgrading do the following:
 - For Active-Active databases, run the following command from one of the clusters:


```
crdb-cli crdb update --crdb-guid <CRDB-GUID> --force
```
 - For Active-Passive (Replica Of) databases: use the admin console to verify that the destination syncer has the correct certificate for the source proxy (DMC). For details, see [Configure TLS for Replica Of](#).

Issues resolved in build 96

- RS67133 - An issue in Redis Enterprise Software affected replication in replica databases using RedisGraph, RedisSearch, and RedisGears in specific scenarios. The problem appeared when importing an RDB file or while synchronizing target Active-Passive (ReplicaOf) databases.

This issue is fixed in Redis Enterprise Software v6.2.10-96 and RedisGraph v2.8.11. We recommend upgrading to these versions at your earliest opportunity. (Failure to upgrade can lead to data loss.)

Once the upgrades are complete, secondary shards might need to be restarted. You can use `rlutil` to restart secondary shards:

```
rlutil redis_restart redis=<shard-id1>,<shard-id2>,...
```

Issues resolved in build 100

- RS74171 - A new command was added as part of Redis 6.2: [XAUTOCLAIM](#). When used in an Active-Active configuration, this command may cause Redis shards to crash, potentially resulting in data loss. The issue is fixed in Redis Enterprise Software version 6.2.12. Additionally, we recommend enabling AOF persistence for all Active-Active configurations.

Issues resolved in build 121

- RS68668, RS72082 - Improvements for internode encryption certification rotation
- RS72304 - Avoid starting a master shard when both master and replica shards crash and the replica did not finish recovery
- RS74469 - Fix for some Redis Active-Active + Redis Streams scenarios that could lead to shard crash during backup; failure to backup

Issues resolved in build 129

- RS77003 - Add grace time to job scheduler to allow certificate rotation in case of failure due to scheduling conflicts.
- RS71112 - Update validation during db configuration to not fail due to ports associated with nodes that are no longer in the cluster. This was done to allow db configuration during adding and removing nodes as part of load balancing.

- RS78486 - Fix known issue from 6.2.10 build 100 - When using rladm tune db to change the replica buffer size, the command appears to succeed, but the change does not take effect.

Known limitations

- RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.
- RS78364 - When using rladm tune db to change the replica buffer size, the command appears to succeed, but the change does not take effect. This issue was introduced in build 100; it will be fixed in a future build of Redis Enterprise Software v6.2.10 and in the next release (v6.2.12).
- RS63258 - Redis Enterprise Software is not currently supported on RHEL 8 with FIPS enabled.

FIPS changes system-generated keys, which can limit secure access to the cluster or the admin console via port 8443.

- RS63375 - RHEL 7 clusters cannot be directly upgraded to RHEL 8 when hosting databases using modules.

Due to binary differences in modules between the two operating systems, you cannot directly update RHEL 7 clusters to RHEL 8 when those clusters host databases using modules. Instead, you need to create a new cluster on RHEL 8 and then migrate existing data from your RHEL 7 cluster. This does not apply to clusters that do not use modules.

All [known limitations](#) listed in the v6.2.4 release notes have been addressed.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory /etc/opt/redislabs/ even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Known issues

- The ZRANGESTORE command, with a special zset-max-ziplist-entries configuration can crash Redis 6.2. See [Redis repository 10767](#) for more details.
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use [rladmin cluster config](#) to configure handle_redirects or handle_metrics_redirects.

Security

Open Source Redis Security fixes compatibility

As part of Redis commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with open source Redis. The following [Open Source Redis CVE's](#) do not affect Redis Enterprise:

- [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16) security fixes for [recent CVE's](#). Redis Enterprise has already included the fixes for the relevant CVE's. Some CVE's announced for Open Source Redis do not affect Redis Enterprise due to different and additional functionality available in Redis Enterprise that is not available in Open Source Redis.

Redis Enterprise Software Release Notes 6.2.8 (October 2021)

Redis Enterprise Software version 6.2.8 is now available!

Features and enhancements

This version features:

- Support for Red Hat Linux Edition (RHEL) 8
- You can now set the start time for [12- and 24-hour backups](#)
- Compatibility with version of [open source Redis 6.2.3](#) (starting with [Redis Enterprise Software v6.2.4](#))
- Compatibility with the security fixes of the latest [open source Redis 6.2.6](#)
- Enhancements and bug fixes

Version changes

Prerequisites and notes

- You can [upgrade to v6.2.8](#) from Redis Enterprise Software v6.0 and later.
- Refer to the [v6.2.4 release notes](#) for important notes regarding the upgrade process.
- When upgrading a cluster from Redis Enterprise 6.0.8 and earlier to 6.2.8 only, the DMC proxy might crash when proxy certificates contain additional text as comments. Redis removes these comments during upgrade, but a change to the v6.2.8 internal upgrade action sequence might cause this problem.

If you plan to upgrade from a pre-6.0.8 release to 6.2.8, check whether your proxy certificate includes additional comments and manually remove them. The change was reverted in 6.2.10.

- Upgrades from versions earlier than v6.0 are not supported.

Product lifecycle updates

As of 31 October 2021, Redis Enterprise Software v5.6.0 is end of life (EOF).

To learn more, see the Redis Enterprise Software [product lifecycle](#), which details the release number and the end-of-life schedule for Redis Enterprise Software.

Redis Enterprise modules have individual release numbers [and lifecycles](#).

Redis modules

Redis Enterprise Software v6.2.8 includes the following Redis modules:

- [RediSearch v2.0.11](#)
- [RedisJSON v1.0.8](#)
- [RedisBloom v2.2.6](#)
- [RedisGraph v2.4.7](#)
- [RedisTimeSeries v1.4.10](#)

To learn more, see [Upgrade the module for a database](#).

Resolved issues

User interface fixes

- RS58804 - Display an error message in case of a login attempt with an LDAP user
- RS56680 - Notify that SASLAUTHD should be disabled prior to enabling LDAP
- RS55844 - Use the correct password and mask it on LDAP password update
- RS60877 - Fixed reset of Active-Active database compression level, in cases where the compression level wasn't set to default, when changing any other configuration via the DB configuration page
- RS43999 - Fixed UI database configuration to allow changes when SFTP SSH key is customized
- RS59861 - Fixed the UI to display an explanation error message when password complexity does not meet requirements
- RS57734 - Fixed inaccessible UI after cluster upgrade due to missing certificate

- RS43041 - Mask secret keys for backup destination for view and edit in the UI

Additional fixes

- RS60068 / RS59146 - Fixed unresolved endpoint due to PDNS issues
- RS52812 - Expand API wrapper to return API 405 errors as JSON/XML
- RS57666 - Fixed false shard migration message when the shard fails to bind the port
- RS57444, RS55294, RS4903 - Fixed false “backup finished successfully” message when the backup failed due to restricted access to the backup destination

Fixes on build #53

- RS67829 - Fixed a bug that caused Modules' auxiliary field not to get replicated between the primary and the replica shards. Applicable for RediSearch, RedisGraph and RedisGears and happening only at following scenarios: - (A) On the destination databases of a Replica Of upon a full sync operation - (B) Upon import operation

Known limitations

-RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.

- RS63258 - Redis Enterprise Software 6.2.8 is not supported on RHEL 8 with FIPS enabled.
FIPS changes system-generated keys, which can limit secure access to the cluster or the admin console via port 8443.
- RS63375 - RHEL 7 clusters cannot be directly upgraded to RHEL 8 when hosting databases using modules.

Due to binary differences in modules between the two operating systems, you cannot directly update RHEL 7 clusters to RHEL 8 when those clusters host databases using modules. Instead, you need to create a new cluster on RHEL 8 and then migrate existing data from your RHEL 7 cluster. This does not apply to clusters that do not use modules.

All [known limitations](#) from v6.2.4 have been fixed.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Known issues

- A new command was added as part of Redis 6.2: [XAUTOCLAIM](#). When used in an Active-Active configuration, this command may cause Redis shards to crash, potentially resulting in data loss. The issue is fixed in Redis Enterprise Software version 6.2.12. Additionally, we recommend enabling AOF persistence for all Active-Active configurations.
- The `ZRANGESTORE` command, with a special `zset-max-ziplist-entries` configuration can crash Redis 6.2. See [Redis repository 10767](#) for more details.
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use `radmin cluster config` to configure `handle_redirects` or `handle_metrics_redirects`.

Security

Open source Redis security fix compatibility

As part of its commitment to security, Redis Enterprise Software implements the latest [security fixes](#) available with open source Redis.

The following [open source Redis CVEs](#) do not affect Redis Enterprise:

- [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
- [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

- [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

Some CVEs announced for Open Source Redis do not affect Redis Enterprise due to functionality that is either different from (or not available in) open source Redis.

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.2.4 (August 2021)

Redis Enterprise Software version 6.2.4 is now available!

This version offers:

- Encryption of all communications within cluster nodes
- Security enhancements
- Bug fixes
- Compatibility with the latest version of open source Redis 6.2.3

Version changes

Prerequisites and notes

You can [upgrade to v6.2.4](#) from Redis Enterprise Software v6.0 and later.

Keep the following in mind:

- Upgrades from versions earlier than v6.0 are not supported
- The new internode encryption feature requires port 3342 to be open on all machines in the cluster.
- In [v6.0.20](#), Redis Enterprise Software replaced Nginx with envoy to improve internal security and communication. As of v6.2.4, Nginx is no longer provided with Redis Enterprise Software.

Database upgrade default changes

The default behavior of the `upgrade db` command has changed. It is now controlled by a new cluster policy (`redis_upgrade_policy`), which defines the policy for creating new databases and upgrading existing databases. The policy supports the following values:

- When set to `major`, the policy allows databases to be created or updated to versions of Redis compatible with open source Redis major releases. This allows for longer upgrade cycles by supporting Redis versions across multiple Redis Enterprise Software releases.

This is the default value for Redis Enterprise Software.

- When set to `latest`, the policy creates new databases and upgrades existing ones to be compatible with the latest (most recent) version of open source Redis, which was the default behavior of earlier versions of Redis Enterprise Software. This is no longer the default behavior.

Setting the upgrade policy to `latest` ensures that the most recent Redis features are available to new databases and ones that are upgraded. It also requires more frequent upgrades, as open source Redis is updated more frequently than Redis Enterprise Software.

The Redis Enterprise Software 6.2.4 package includes compatibility with the most recent major Redis release (v6.0) and the latest (most recent) update to Redis (v6.2.3).

By default, compatibility with v6.0 will be installed. To change this, use `r1admin` to set the upgrade policy and the default Redis version:

```
$ r1admin tune cluster redis_upgrade_policy latest  
$ r1admin tune cluster default_redis_version 6.2
```

To learn more, see the [upgrade instructions](#).

Product lifecycle updates

Redis Enterprise Software v5.6.0 will reach end of life (EOF) on October 31, 2021.

To learn more, see the Redis Enterprise Software [product lifecycle](#), which details the release number and the end-of-life schedule for Redis Enterprise Software.

Redis Enterprise modules have individual release numbers [and lifecycles](#).

Deprecation notices

- In [v6.0.20](#), the SASL-based LDAP mechanism was deprecated in favor of a new [RBAC-based approach](#). As of [v6.2.12](#), support for the older mechanism has been removed.
For help migrating to the LDAP-based mechanism, see [Migrate to role-based LDAP](#).
- [OpenStack Object Storage](#) ("Swift") has reached end-of-life. Consequently, you can no longer use ObjectStack Swift as a target for database backup or export operations.

Features and enhancements

Internode encryption

Internode encryption (INE) encrypts all communication between nodes in a cluster; it is available for the control plane and the data plane.

Control plane internode encryption

Control plane internode encryption encrypts all management communication within a cluster. It is enabled by default for all new clusters and upgraded clusters.

Data plane internode encryption

Data plane internode encryption encrypts communication between nodes within a cluster, such as database replication between nodes.

Data plane internode encryption is available for new or fully upgraded clusters. It is not enabled by default.

You can enable data plane internode encryption by:

- Setting the cluster policy to enable data plane internode encryption by default for new databases

```
rladmin tune cluster data_internode_encryption enabled
```

- Enabling it for individual existing databases

```
rladmin tune db <db:id | name> data_internode_encryption enabled
```

Internal certificate management

Internode encryption relies on internal certificates signed by a unique, private CA certificate created for your deployment. The private CA generates and signs leaf certificates for internode encryption only. It's generated when you install or upgrade to Redis Enterprise 6.2.4. It's used only within the cluster and is not exposed outside of the cluster.

The leaf certificates expire regularly; they're automatically rotated before expiration and alerts are issued as needed.

Open source Redis compatibility

[Redis 6.2](#) introduced new commands, feature improvements, and security fixes; it addresses many customer requests.

Redis Enterprise Software supports all new commands, except [RESET](#) and [FAILOVER](#). (Redis Enterprise takes a different approach to connectivity; it also separates control plane operations from data plane operations.)

To learn more, see Redis Enterprise Software [compatibility with open source](#).

Redis modules

Redis Enterprise Software v6.2.4 includes the following Redis modules:

- [RediSearch v2.0.11](#)
- [RedisJSON v1.0.8](#)
- [RedisBloom v2.2.6](#)
- [RedisGraph v2.4.7](#)
- [RedisTimeSeries v1.4.10](#)

Internode encryption for modules

To utilize data plane encryption for existing databases with modules, update the module to the latest version prior to enabling data plane encryption.

For help, see [Upgrade the module for a database](#).

Module-related enhancements

Added the capability to update current module arguments for an existing database. In earlier versions, you could do this only when upgrading a module. To learn more, see [rladmin upgrade](#).

Resolved issues

- RS39954 - Changed the UI status indication for the [default user](#) from Active/Inactive to Enabled/Disabled
- RS42626 - Increased the max length for modules commands from 23 characters to 64 characters
- RS54732 - Fixed incorrect reporting of number database connections, which caused the number of connections to be reported as a 20 digit number
- RS52265 - Fixed excessive log lines reporting when an Active-Active database is on featureset 0. [Upgrade the featureset](#) version to the latest.
- RS56122 - Fixed a bug that was causing AOF files to grow when the replicas of two Active-Active databases became disconnected during full synchronization
- RS58184 - Fixed a bug when trying to create an Active-Active database with expired syncer certificates; participating clusters were creating replicas even though the create operation failed.
- RS48988 - Add the username description in the log upon an unauthorized REST API request

Known limitations

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Known issues

- A new command was added as part of Redis 6.2: [XAUTOCLAIM](#). When used in an Active-Active configuration, this command may cause Redis shards to crash, potentially resulting in data loss. The issue is fixed in Redis Enterprise Software version 6.2.12. Additionally, we recommend enabling AOF persistence for all Active-Active configurations.
- The `ZRANGESTORE` command, with a special `zset-max-ziplist-entries` configuration can crash Redis 6.2. See [Redis repository 10767](#) for more details.
- RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.
- RS40641 - API requests are redirected to an internal IP in case the request arrives from a node which is not the master. To avoid this issue, use `rladmin cluster config` to configure `handle_redirects` or `handle_metrics_redirects`.

Security

- The following [Open Source Redis CVE's](#) do not affect Redis Enterprise:
 - [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)

- [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
- [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.0.20 (April 2021)

Redis Enterprise Software version 6.0.20 is now available! This version includes the following new features and improvements:

- A new integration for [LDAP](#) authentication and authorization into RS role-based access controls (RBAC). You can now use LDAP to authorize access to the admin console and to authorize database access.
- An enhanced [clients mutual authentication](#) mechanism, adding the ability to authenticate client connections using a Certificate Authority (CA).
- Support of [Redis eviction policies](#) on [Active-Active](#) Redis databases.
- A new [migration process for Active-Active](#) Redis database using the Active-Passive (Replica Of) mechanism.
- Support for the [BITFIELD](#) data type on [Active-Active](#) Redis databases.

And other functional and stability improvements.

Version information

Upgrade instructions

- Follow [these instructions](#) for upgrading to Redis Software 6.0.20 from Redis Software 5.6.0 and above.
 - Note that upgrades from earlier Redis Software versions are not supported.
- For Active-Active deployments, this release requires that you [upgrade the CRDB featureset version](#).
- Upgrades of Active-Active databases to Redis Software 6.0.20, will require all their instances to run with protocol version 1 and featureset version 1 or above. Active-Active databases running on protocol version 0 and/or featureset version 0 will block the upgrade.

Product lifecycle information

- End of Life (EOL) for Redis Enterprise Software 6.0 and earlier versions, can be found [here](#).
- EOL for Redis modules can be found [here](#).

Deprecation Notice

- Upgrades to the next Redis Software will be enabled from version 6.0 and above.
- Support for the SASL-based LDAP mechanism was deprecated in v6.0.20. As of v6.2.12, support has been removed and the feature is obsolete.
- Starting with Redis Software version 6.0.12, Envoy replaces Nginx for internal cluster administration. Support for Nginx is considered deprecated, it will be removed in a future version.

New Features

New LDAP integration

Redis Enterprise Software integrates Lightweight Directory Access Protocol (LDAP) authentication and authorization into its [role-based access controls \(RBAC\)](#). You can now use LDAP to authorize access to the admin console and to manage database access.

Clients Mutual TLS authentication using a Certificate Authority (CA)

Redis Enterprise Software adds the ability to use a Certificate Authority (CA) for client authentications, allowing clients to rotate their certificates without the need to load new certificates to your database.

Redis eviction policies on Active-Active Redis databases

All Redis [eviction policies](#) are now supported on Active-Active Redis databases. You can create new Active-Active databases or edit existing ones using the UI console to enable it.

- Note that eviction is not supported yet for Active-Active Redis databases running with Auto Tiering .

Migration to an Active-Active Redis database

Redis Enterprise Software adds the ability to easily [migrate your Redis database to an Active-Active](#) Redis database using the Active-Passive (Replica Of) mechanism.

BITFIELD on Active-Active Redis databases

Redis Enterprise Software adds the ability to use the BITFIELD data type on Active-Active Redis databases. Please read more about [developing for Active-Active with BITFIELD](#) to understand the conflict resolution and limitations.

Redis modules

The following GA releases of Redis modules are bundled with Redis Software 6.0.20: (Please read the below updates for 6.0.20-97)

- [RediSearch](#), version 2.0.6
- [RedisJSON](#), version 1.0.7
- [RedisGraph](#), version 2.2.14
- [RedisTimeSeries](#), version 1.4.8
- [RedisBloom](#), version 2.2.4

To use the updated modules with a database, you must [upgrade the module on the database](#).

Additional capabilities

- Redis Software 6.0.20 includes open source Redis 6.0.9. For more information about Redis 6.0.9, check out the [release notes](#).
- Redis Software 6.0.20 adds new `radmin` commands for setting Ciphers suites and minimal TLS version for:
 - Control plane: setting Envoy
 - Data plane: setting the Proxy for clients connections
 - Sentinel discovery service
- Starting with Redis Software 6.0.20, new clusters will be set with minimal TLS version v1.2

All known bugs around setting ciphers were fixed. To learn more, see [Configure cipher suites](#).

- Starting with Redis Software 6.0.20, the [syncer process](#) was improved to automatically recover and resume synchronisation after reaching out-of-memory.
- Envoy updated and verified with multiple security headers.
- Starting with Redis Software 6.0.20:
 - The replication backlog size of new databases is allocated dynamically according to shard size.
 - The Active-Active replication backlog size of new Active-Active databases is allocated dynamically according to shard size.

Important fixes

- RS50905, RS54809, 54940 - Fix in Redis preventing missing process PID
- RS53639 - Fix to avoid stuck state machine when assigning incorrect Redis ACL with the allkeys alias or ~* and also with ~<somekey>
- RS47983 - Fixed dependencies with installation using custom directories
- RS54382 - Fixed missing API documentation
- RS52433, RS53417, RS42195, RS42194, RS30526, RS46821, RS48928 - Fixed security headers for Envoy

Starting 6.0.20-69

- RS55504 - Fixed issue that caused RediSearch cursor to break.

Starting 6.0.20-97

- RS57659 - Fixed force removing an Active-Active instance which is hosted on an inaccessible cluster
- RS57315 - Fixed creation and editing of an Active-Active database with LDAP users of the new RBAC LDAP integration. This applies to UI access and REST API and does not apply to LDAP users for database access via Redis Clients
- RS57073 - Fixed a bug caused in the shard migration process which could leave unattended shards on the node
- RS56508 - Fixed backwards compatibility of client certificate when upgrading from earlier versions and using a certificate chain with Extended Key Usage extension being set to "TLS Web Server Authentication" instead of "TLS Web Client Authentication"
- RS49289 - Fixed updating the log rotation config file according to the custom config path that was set during the installation
- The bundled RedisGraph module was upgraded to v2.4.6
- The bundled RedisTimeSeries module was upgraded to v1.4.9
- The bundled RediSearch module was upgraded to v2.0.8

Known limitations

-RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.

- RS59983 - Clients may get disconnected by the proxy when one client sends an UNSUBSCRIBE command without being subscribed to any channel and disconnect before the response returns back from the server (from the proxy).
- RS60068 - The pdns might not resolve the master node after master node change. Restarting the pdns service is required in this case.
- RS61114 - Active-Active synchronization will fail in the following scenario: a new syncer connection is established AND a partial sync (psync) was initiated AND a cron job runs before the first ACK of the psync was received.
- RS55504 - Bug RS6.0.20-66 (Build #66) causes RediSearch cursor to break. Please upgrade to a higher build when running with RediSearch.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory /etc/opt/redislabs / even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Upgrade

- [Redis Software 5.4.2](#) introduced new Active-Active Redis Database capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis Database is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the Active-Active Redis Database protocol](#).
- When you upgrade an Active-Active Redis with active AOF from [Redis Software 5.4.2](#) or earlier to version [Redis Software 5.4.4](#) or later:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For help with this issue, contact Support.
- Starting from [Redis Software 5.4.2](#), to preserve the current Redis major.minor version during database upgrade you must use the keep_redis_version option instead of keep_current_version.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis modules.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE commands for string data-type in CRDBs return “string” (OSS Redis standard).

Security

- As part of Redis commitment to security, the following [Open Source Redis CVE's](#) have been addressed in Redis Enterprise 6.0.20:
 - [CVE-2021-32626](#) - Lua scripts can overflow the heap-based Lua stack. This has been addressed in Redis Enterprise 6.0.20-62
 - [CVE-2021-32627](#) - Integer overflow issue with Streams. This has been addressed in Redis Enterprise 6.0.20-1
 - [CVE-2021-32628](#) - Vulnerability in handling large ziplists. This has been addressed in Redis Enterprise 6.0.20-1
 - [CVE-2021-32687](#) - Integer overflow issue with intsets. This has been addressed in Redis Enterprise 6.0.20-89
- The following [Open Source Redis CVE's](#) do not affect Redis Enterprise:
 - [CVE-2021-32625](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis since Redis Enterprise does not implement LCS. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.4, Redis 6.0.14)
 - [CVE-2021-32672](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the LUA debugger is unsupported in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
 - [CVE-2021-32675](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proxy in Redis Enterprise does not forward unauthenticated requests. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
 - [CVE-2021-32762](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the memory allocator used in Redis Enterprise is not vulnerable. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)
 - [CVE-2021-41099](#) - Redis Enterprise is not impacted by the CVE that was found and fixed in open source Redis because the proto-max-bulk-len CONFIG is blocked in Redis Enterprise. Additional information about the open source Redis fix is on [the Redis GitHub page](#) (Redis 6.2.6, Redis 6.0.16)

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.0.12 (January 2021)

Redis Enterprise Software (RS) 6.0.12 is now available! This version includes the following new features and improvements:

- Synchronization can now be [distributed across the nodes](#) of Active-Active or Active-Passive databases
- You can [disable several internal RS services](#) to free up more memory
- User accounts can have multiple passwords to allow for [password rotation](#)
- Dependencies are automatically installed when you add modules to a cluster
- Envoy replaces NGINX for internal cluster administration
- Automatic recovery of the [syncer process](#) from out-of-memory (preview mode)

And other functional and stability improvements.

Version information

Upgrade instructions

- Follow [these instructions](#) for upgrading to RS 6.0.12 from RS 5.4.0 and above.
- For Active-Active deployments, this release requires that you [upgrade the CRDB featureset version](#).

Product lifecycle information

- End of Life (EOL) for Redis Enterprise Software 6.0 and previous RS versions, can be found [here](#).

- EOL for Redis modules can be found [here](#).

Deprecation Notice

- Support for RS 5.4.X ended on December 31, 2020.
- Support for Red Hat Enterprise Linux 6 and Oracle Linux 6 and Ubuntu 14.04 (Trusty) operating systems platforms ended on November 30, 2020.
- This is the last RS version that supports direct upgrades from versions below 5.6.0.
- This is the last RS version that supports Active-Active protocol version below 1 and featureset version below 1.

New Features

Distributed Syncer

The syncer process now supports running in a [distributed mode](#). This option can improve the latency for Active-Active databases with a very high throughput profile. You can configure a replicated database to use distributed synchronization so that any available proxy endpoint can manage synchronization traffic.

Disabling RS services to free memory

Redis Software users can now use the REST API to [disable the following services](#):

- cm_server
- mdns_server
- pdns_server
- stats_archiver
- saslauthd
- crdb_coordinator
- crdb_worker

Once disabled, services are not monitored and controlled by the supervisord.



Warning - This feature can cause unintended results if the cluster relies on the disabled services. To make sure you understand the impact of disabled services, test the system in a lab environment before you deploy in production.

Support for multiple passwords

For users of Redis 6 and RS 6.0 and above, you can now add more security to your password management by maintaining multiple passwords for a user to allow seamless password rotation.

As of RS 6.0, you can assign specific data access permissions (Redis ACLs) and cluster administration permissions to users. Password rotation is especially helpful so that you can do a rolling update of the passwords in the application clients that connect to the Redis databases.

In this version, you can only configure multiple passwords [using the REST API](#).

Redis Modules dependencies management

RedisGears GA and RedisAI GA require Redis Software to fetch and manage external dependencies. Modules declare dependencies at release time in their ramp file.

In this version of RS, these dependencies are installed by Redis Software when the module is added to the cluster. The master node downloads the required dependencies and prompts the other nodes to copy the dependencies from the master node. When all dependency requirements are satisfied, the module installation is complete. New nodes to the cluster also automatically install the dependencies.

Syncer automatic recovery from out-of-memory (Preview mode)

For Active-Active databases, the syncer process gracefully recovers from an out of memory (OOM) state.

Although Active-Active synchronization is bi-directional, in each direction we can define a source instance and a destination instance. When a destination instance (that is, the instance running the syncer process) gets OOM, the syncer process attempts to automatically recover the data synchronization when memory becomes available.

This is a configurable option and currently under preview mode. This behavior will be GA and set as default in the next RS version.

To enable the syncer automatic recovery, do these steps on each participating cluster:

1. [Upgrade the featureset version](#) to 3.
2. Enable the syncer automatic recovery using the REST API:

```
curl -v -k -u <username>:<password> -X PUT -H "Content-Type: application/json" -d
'{crdt_syncer_auto_oom_unlatch":true}' http://<cluster_address>:8080/v1/b dbs/<database_ID>
```

The syncer process restarts to with automatic recovery on.

Redis modules

The following GA releases of Redis modules are bundled with RS 6.0.12:

- [RedisSearch](#), version 2.0.6
- [RedisJSON](#), version 1.0.4
- [RedisGraph](#), version 2.2.11
- [RedisTimeSeries](#), version 1.4.7
- [RedisBloom](#), version 2.2.4

To use the updated modules with a database, you must [upgrade the module on the database](#).

Additional capabilities

- RS 6.0.12 includes open source Redis 6.0.6. For more information about Redis 6.0.6, check out the [release notes](#).
- The bundled Nginx version was updated from version 1.16.0 to 1.18.0.
- The crdb-cli syntax to remove an instance is changed from `remove-instance [--ordered|--unordered]` to `remove-instance [--force|--no-force]`.

Important fixes

- RS45627, RS47382 - Fixed bugs causing clients to disconnect when using XREAD and XREADGROUP commands in blocking mode on other clients' connections.
- RS44656 - Fixed a bug causing TLS mode for clients connections to toggle between 'all communication' to 'for crdb communication only' when performing a global configuration change.
- RS42587 - Fixed a bug in the web UI console if the FQDN of the cluster is a substring of the FQDN of a participating cluster
- RS49404 - Fixed a bug in for upgrades with custom directories that prevent users from creating databases via the web UI console.
- RS43961 - bigkeys command fixed to handle non-printable key names
- RS45707 - Fixed a bug that caused RCP (Redis Cloud Pro) databases to reject connections while resharding the database.
- RS51144 - Fixed a bug in the syncer process that was stopping synchronization between all instances in some scenarios of network disconnection of one or more participating clusters.

with 6.0.12-58:

- RS50865 - Fixed a bug causing radmin change master node to fail when performed after a prior successful master change.
- RS51359 - Fixed a memory leak on replica shards in Active-Active databases with replication and AOF for persistence.
- RS52363 - Updated PUB/SUB max message value size from 64KB to 512MB

Known limitations

-RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs/` even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Upgrade

- RS 5.4.2 introduced new Active-Active Redis Database capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis Database is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the Active-Active Redis Database protocol](#).
- When you upgrade an Active-Active Redis with active AOF from version RS 5.4.2 or earlier to version RS 5.4.4 or later:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.

- If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, contact Support.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the keep_redis_version option instead of keep_current_version.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis modules.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE commands for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.0.8 (September 2020)

Redis Enterprise Software (RS) 6.0.8 is now available! This version includes the new RediSearch 2.0 module, open source Redis 6.0.5, changes the rladmin tool for upgrading modules, and includes bug fixes.

Version information

Upgrade instructions

Follow [these instructions](#) for upgrading to RS 6.0.8 from RS 5.4.0 and above. For Active-Active deployments, this release requires that you [upgrade the CRDB featureset version](#).

End of life

End of Life (EOL) for Redis Enterprise Software 6.0 and previous RS versions, can be found [here](#). EOL for Redis Modules can be found [here](#).

- Support for Red Hat Enterprise Linux 6 and Oracle Linux 6 [operating systems platforms](#) will end on November 30, 2020.
- Support for Ubuntu 14.04 (Trusty Tahr) [operating systems platforms](#) will end on November 30, 2020.

New features

Open source Redis 6

RS 6.0 includes open source Redis 6.0.5. For more information about Redis 6.0.5, check out the [release notes](#).

Upgrading Redis modules via rladmin

The [rladmin CLI](#) introduces several updates to the commands for upgrading modules. It is now easier to upgrade your modules to the latest module version. Find out more [here](#).

Redis modules

The following GA releases of Redis Modules are bundled in RS 6.0:

- [RediSearch](#), version 2.0 (updated)
- [RedisJSON](#), version 1.0.4
- [RedisGraph](#), version 2.0.19 (updated)
- [RedisTimeSeries](#), version 1.2.7 (updated)
- [RedisBloom](#), version 2.2.4 (updated)

To use the updated modules with a database, you must [upgrade the module on the database](#).

Additional capabilities

- [Shard level metrics](#) have been added to the metrics_exporter and are now available from Prometheus. You can find all of the metrics [here](#).
- RS DEB packages (for Ubuntu) and RPM packages (for RHEL) are now signed with a GPG key so customers can verify that the package is authentic and

has not been tampered with. You can access the GPG on the [installaion page](#).

- The [crdb-cli](#) history log is now being added to support packages.

Important fixes

- RS33193 - Improved log files handling in the proxy for large files.
- RS43572 - Fixed a bug causing the UI to fail when enabling SMTP STARTTLS.
- RS46062 - Fixed missing metrics of Active-Active databases in Grafana.
- RS44758 - Fixed non responding button for saving a new user via the UI. With build 6.0.8-32:
- RS45627, RS47382 - Fixed bugs causing clients to disconnect when using XREAD and XREADGROUP commands in blocking mode on other clients' connections.

Known limitations

-RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering . Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.

Active-Active databases

- RS44656 - A bug causing TLS mode for clients connections to toggle between 'all communication' to 'for crdb communication only' when performing a global configuration change. **TBD**
- RS51359 - Active-Active databases, using replication and Append Only File (AOF) for [Database persistence](#), are suffering from memory leaks on replica shards, causing them to grow bigger than the master shards. Customers are advised to upgrade to RS 6.0.12 **TBD**. Meanwhile you can use snapshots for database persistence or restart the replica shards **TBD**.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory `/etc/opt/redislabs` / even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount `/etc/opt/redislabs` to one of the custom, non-root directories.

Upgrade

- **RS 5.4.2** introduced new Active-Active Redis Database capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis Database is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the Active-Active Redis Database protocol](#).
- When you upgrade an Active-Active Redis with active AOF from version **RS 5.4.2** or lower to version **RS 5.4.2** or higher:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, contact [Support](#).
- Starting from **RS 5.4.2**, to preserve the current Redis major.minor version during database upgrade you must use the `keep_redis_version` option instead of `keep_current_version`.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- Starting from RS 5.4.2 and after you upgrade an Active-Active database, TYPE commands for string data-type in Active-Active databases return "string" (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 6.0 (May 2020)

Redis Enterprise Software (**RS**) **6.0** is now available! This new version bundles open-source Redis 6, implements enhanced Access Control List (ACL) capabilities using Role-Based Access Control (RBAC) for database access, and adds the support of Redis Streams on Active-Active databases.

Version information

Upgrade instructions

Follow [these instructions](#) for upgrading to RS 6.0 from RS 5.4.0 and above. For Active-Active deployments, this release requires that you [upgrade the CRDB featureset version](#).

End of life

End of Life (EOL) for Redis Enterprise Software 6.0 and previous RS versions, can be found [here](#). EOL for Redis Modules can be found [here](#).

- Support for Red Hat Enterprise Linux 6 and Oracle Linux 6 [operating systems platforms](#) will end on November 30, 2020.
- Support for Ubuntu 14.04 (Trusty Tahr) [operating systems platforms](#) will end on November 30, 2020.

New features

Open source Redis 6

RS 6.0 bundles latest open source Redis 6. For more information, check out the [Diving into Redis 6](#) article.

Access control list (ACL)

Based on OSS Redis 6, RS 6.0 offers the ability to manage and control connections to your databases using users and their data access permissions in terms of commands they can execute and keys they can access.

In OSS Redis, the ACLs are managed separately per user for each database. In Redis Enterprise Software, Redis ACLs are managed for the databases at the cluster. For more information, check out the [Redis Enterprise Software user management documentation](#).

Role-based access control (RBAC)

RS 6.0 leverages Redis ACLs to implement role-based access control that easily scale and manage data access permissions. Using roles minimizes the overhead involved in managing a cluster with many databases, multiple users, and various access control lists. For more information, check out the [Redis Enterprise Software user management documentation](#).

Active-Active support for Redis Streams

RS 6.0 adds support for Redis streams on Active-Active geo-distributed databases using conflict-free replicated data type (CRDT). You can now use all Redis streams commands including consumer groups on Active-Active databases. To enable it, [upgrade your Active-Active database featureset version](#) to the latest (featureset version = 2) as part of the upgrade process. For more information, check out [Redis Streams on Active Active databases](#).

Redis modules

The following GA releases of Redis Modules are bundled in RS 6.0:

- [RedisBloom](#), version 2.2.2 (updated)
- [RedisGraph](#), version 2.0.11 (updated)
- [RedisJson](#), version 1.0.4
- [RediSearch](#), version 1.6.12 (updated)
- [RedisTimeSeries](#), version 1.2.5 (updated)

To use the updated modules with a database, you must [upgrade the module on the database](#).

Additional capabilities

- Added the ability to configure a storage service that uses the S3 protocol. Configuration for [backup location](#) and for [import](#) and [export](#) locations of [RDB files](#) is possible. The storage service must have a valid SSL certificate. To connect to an S3-compatible storage location, run: `radmin cluster config s3_url <url>`
- The crdb-cli tool was updated so it is now displaying the Active-Active database's featureset version and the protocol version per instance. For example:

```
$ crdb-cli crdb list --verbose
  CRDB-GUID          NAME      REPL-ID PROTOCOL   FEATURESSET    DB-ID      CLUSTER-FQDN
969122be-... lorem ipsum1      1        0        bdb:1      cluster1.local
969122be-... lorem ipsum1      2        0        bdb:1      cluster2.local
```

- Added REST API and radmin commands to modify the timeout for [automatically disconnecting an inactive admin console session](#).
 - Using the radmin run: `radmin cluster config cm_session_timeout_minutes <int_value>`

- Using the REST API:

```
curl --request PUT \
--url https://localhost:9443/v1/cluster \
--header 'content-type: application/json' \
--data '{
    "cm_session_timeout_minutes": 10
}'
```

- Added no_of_expires metrics for database metrics and for shard metrics. You can access this metric from the REST API:
 - no_of_expires shows the current number of volatile keys in the database.
 - expired_objects shows the rate of keys expired in DB (expirations/sec).
- Added the ability to customize the welcome message on the login page in the admin console console.

Important fixes

- RS39121, RS35335 - Optimized the XREAD and XREADGROUP commands so when using them in a non blocking fashion (without BLOCK keyword) they use the shared connection instead of a dedicated connection.
- RS26448 - Fixed 'ram overhead' metric calculation for ROF databases using AOF.
- RS31190 - Fixed a bug that causes databases with OSS Cluster API enabled to override the 'preferred IP type' attribute from 'external' to 'internal' (the default value).
- RS34009 - Updated the modules loading procedure for clusters with FIPS compliance enabled.
- RS38233 - Improved the Redis cleanup job handling the persistent directory
- RS39228 - Fixed a bug in the WAIT command that in some cases was released after a longer period than requested.
- RS39749 - Fixed a bug that blocked eviction while LUA scripts were in progress.
- RS43996 - Fixed a bug when aborting an upgrade to RS 6.0.

Known limitations

-RS81463 - A shard may crash when resharding an Active-Active database with Auto Tiering. Specifically, the shard will crash when volatile keys or Active-Active tombstone keys reside in Flash memory.

Active-Active databases

- RS51359 - Active-Active databases, using replication and Append Only File (AOF) for [database persistence](#), are suffering from memory leaks on replica shards, causing them to grow bigger than the master shards. Customers are advised to upgrade to RS 6.0.12 **TBD**. Meanwhile you can use snapshots for database persistence or restart the replica shards **TBD**.

Installation limitations

Several Redis Enterprise Software installation reference files are installed to the directory /etc/opt/redislabs/ even if you use [custom installation directories](#).

As a workaround to install Redis Enterprise Software without using any root directories, do the following before installing Redis Enterprise Software:

1. Create all custom, non-root directories you want to use with Redis Enterprise Software.
2. Mount /etc/opt/redislabs to one of the custom, non-root directories.

Upgrade

- RS 5.4.2 introduced new Active-Active Redis Database capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis Database is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must upgrade the Active-Active Redis Database protocol.
- When you upgrade an Active-Active Redis with active AOF from version RS 5.4.2 or lower to version RS 5.4.4 or higher:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the keep_redis_version option instead of keep_current_version.
- Dynatrace agent installed on the cluster nodes can hamper the working on Envoy process leading to failure of UI and REST API. Prior upgrading we recommend removing Dynatrace completely or try upgrading to newer versions.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- Starting from RS 5.4.2 and after you upgrade an Active-Active database, TYPE commands for string data-type in Active-Active databases return

"string" (OSS Redis standard).

Updated: August 17, 2023

Previous releases

Version (Release date)	Major changes	OSS Redis compatibility
5.6.0 (April 2020)	Install improvements for RHEL 6 and 7. Active-Active support for HyperLogLog. Auto Tiering now supports RedisJSON. Active-Active default changes for high availability and OSS Cluster API support. Backup support for Google Cloud Storage and Azure Blob storage.	Redis 5.0.8
5.5 preview (April 2019)	Preview release. Databases support multiple modules.	
5.4.14 (February 2020)		
5.4.10 (December 2019)		
5.4.6 (July 2019)		
5.4.4 (June 2019)		
5.4.2 (April 2019)		
5.4 (December 2018)		
5.2.2 (August 2018)		
5.3 beta(July 2018)		
5.2 (June 2018)		
5.0.2 (2018 March)		
5.0 (November 2017)		
4.5 (May 2017)		
4.4 (December 2016)		
4.3.0-230 (August 2, 2016)		
4.2.1-30 (October 18, 2015)		
4.0.0-49 (June 18, 2015)		
0.99.5-24 (February 15, 2015)		
0.99.5-11 (January 5, 2015)		

Updated: March 7, 2023

Redis Enterprise Software Release Notes 5.6.0 (April 2020)

Redis Enterprise Software (RS) 5.6.0 is now available. This major version release includes:

- Improved installation process to be customizable
- Support for the HyperLogLog data type in Active-Active databases
- RedisJSON support in databases with Auto Tiering enabled
- Support for Redis OSS Cluster API for Active-Active and Replica Of databases
- Replica HA enabled by default for Active-Active databases
- Cloud backup locations support
- Support for Red Hat Enterprise Linux version 7.7, 7.8
- Additional enhancements, and minor bug fixes

Information

Upgrade

- Follow [these instructions](#) for upgrading to RS 5.6.0 from RS 5.0.2 and above.
- For Active-Active databases, you must upgrade the [feature set version](#).

End of life

- End of Life (EOL) for Redis Enterprise Software 5.6 and previous RS versions can be found [here](#). EOL for Redis Modules can be found [here](#).
- Support of Red Hat Enterprise Linux 6 and Oracle Linux 6 [operating systems platforms](#) will end on November 30, 2020.
- Support of OpenStack Object Storage (“Swift”) for [backup, import and export location](#) will end on November 30, 2020.

New features

Redis Software installer

Redis Enterprise Software installer adds the ability to specify custom installation paths and a custom installation user, group, or both on RHEL versions 6 and 7.

When you run the installer you can specify the installation, the configuration and the var as well as the user and the group:

```
sudo ./install.sh --install-dir <path> --config-dir <path> --var-dir <path> --os-user <user> --os-group <group>
```

For more information, check out the [Redis Enterprise Software installer documentation](#).

Hyperloglog on Active-Active

RS 5.6.0 adds the support of [HyperLogLog](#) data structure for [Active-Active Redis](#) databases.

Because HyperLogLog is a counting data structure by nature, conflicts can occur when deleting entries. For efficiency, performance and memory considerations, conflicts between instances are resolved with DEL (delete) operations winning over ADD operations that took place in concurrent or before the DEL operation.

For more information, check out the [HyperLogLog on Active-Active documentation](#).

RedisJSON on Auto Tiering

RS 5.6.0 adds support for [Auto Tiering](#) databases with [Redis Modules](#). RedisJSON is the first module to be available to run on a database with Auto Tiering enabled.

Using RedisJSON on Auto Tiering allows you to benefit from high performance redis with high data volume at lower costs.

OSS Cluster API support

For more information, check out the [OSS Cluster API documentation](#).

Support for Active-Active and Replica Of databases

You can configure [Active-Active](#) databases to use the [OSS Cluster API](#) using the admin console. The OSS Cluster API improves the performance of user operations against your database.

You can also create or modify an Active-Active Redis database in OSS Cluster mode using the `crdb-cli` tool with the `--oss-cluster` option to apply the changes to all of the instances.

Create and edit database using the admin console

You can configure OSS Cluster API for databases using the admin console.

For Active-Active databases, you can create the database with OSS Cluster API enabled for all of its instances. When you enable OSS Cluster after the Active-Active database is created, the change applies only to the local instance.

Replica HA defaults for Active-Active

To enhance the availability and the consistency of [Active-Active geo-distributed Redis](#), [Replica HA](#) is enabled by default for all existing and new Active-Active Redis Databases.

To disable Replica HA for for a local instance of an Active-Active database, run this command on the instance: `radmin tune db <bdb_uid> slave_ha disabled`

Cloud backup locations support

Redis Enterprise 5.6.0 adds the ability to configure Azure Blob Storage and Google Cloud Storage as backup, import, and export locations.

Red Hat Enterprise Linux 7.7, 7.8

Redis Enterprise 5.6.0 adds RHEL 7.7, 7.8 to its [supported platforms](#).

Redis modules

The following GA releases of Redis Modules are bundled in RS 5.6.0:

- [RedisBloom](#), version 2.2.1
- [RedisGraph](#), version 2.0.10
- [RedisJson](#), version 1.0.4
- [RedisSearch](#), version 1.6.11 (updated)
- [RedisTimeSeries](#), version 1.2.3

Additional capabilities

- OSS Redis version 5.0.8 is merged into RS 5.6.0.
- Starting RS 5.6.0, to upgrade modules during database upgrade you must use the module_args option instead of keep_module_args or "" (for no arguments). The module_args option must follow with one of the following (Always in quotation marks): "keep_args" or " " or "". For more info and examples, check out [upgrading Redis Modules documentation](#).
- rladmin adds the ability to demote the cluster master node when setting it to [maintenance mode](#) by using the demote_node option:

```
rladmin node <node_uid> maintenance_mode on demote_node
```

- The SENTINEL MASTER command output format was updated to be aligned with OSS equivalent output.

Important fixes

- RS38315, RS33747 - Added the LUA script name (SHA) and its arguments to the warning message indicating the LUA script has been running for more than 5 seconds.
- RS38706 - Fixed a bug which caused a stuck state machine in some scenarios of restoring DB from RDB.
- RS34309 - Improved internal passwords handling.
- RS38498 - Fixed a bug in the upgrade process of a database to avoid failure when saving large RDB (Redis Database Backup) file.
- RS38706 - Fixed a bug that caused a stuck state machine after restoring a snapshot of an Active-Active database.
- RS43996 - Fixed a bug in the upgrade process when using ./install.sh -y and firewalld is not running.
- RS45777 - Fixed a bug that caused clients on a shared connection to the proxy to get disconnected. Disconnections occurred in case a response for a request of an already disconnected client was received. With build 5.6.0-39:
- RS45627, RS47382 - Fixed bugs causing clients to disconnect when using XREAD and XREADGROUP commands in blocking mode on other clients' connections.
- RS44656 - Fixed a bug causing TLS mode for clients connections to toggle between 'all communication' to 'for crdb communication only' when performing a global configuration change.

Known limitations

Active-Active databases

- RS51359 - Active-Active databases, using replication and Append Only File (AOF) for [database persistence](#), are suffering from memory leaks on replica shards, causing them to grow bigger than the master shards. Customers are advised to upgrade to RS 6.0.12. Meanwhile, you can use a snapshot for database persistence or restart the replica shards.

Upgrade notes

- RS 5.4.2 introduced new Active-Active Redis Database capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis Database is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the Active-Active Redis Database protocol](#).
- When you upgrade an Active-Active Redis with active AOF from version RS 5.4.2 or lower to version RS 5.4.4 or higher:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, contact Support.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the keep_redis_version option instead of keep_current_version.

- Google Chrome browser on macOS Catalina requires a self-signed certificate generated after June 2019 to include the extendedKeyUsage field in order to connect to the RS admin console. If you use a self-signed certificate that does not include this field, [update the self-signed certificate](#).

Modules upgrade

- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- Before you upgrade a database with RediSearch Module to Redis 5.0, you must [upgrade the RediSearch Module](#) to version 1.4.2 or above.

Cluster API

- The API for removing a node is updated in [RS 5.4.2](#) or higher. The API call must include json data and the “Content-Type: application/json” header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Discover service

- For [Redis Sentinel \(Discovery Service\)](#), every database name must be unique across the cluster.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE commands for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.5 Preview (April 2019)

Redis Enterprise Software (RS) 5.5 Preview Edition is now available.

RS 5.5 is a preview version that includes all the capabilities of Redis Enterprise 5.4, plus support for creation of Redis databases with multiple modules and support for these modules:

- RediSearch (GA)
- RedisGraph (GA)
- RedisBloom (GA)
- RedisJSON (GA)
- RedisAI (Preview Version)
- RedisTimeSeries (Preview Version)
- RedisGears (Preview Version)

New features

RS 5.5 lets you create Redis databases with multiple Redis modules.

Preview considerations

This preview version is a standalone version and is not supported for production environments. You cannot upgrade to it from a lower version or upgrade from it to a higher version. Unexpected behaviors/issues found in this release will be addressed in future releases.

This preview version is not supported for networks that are isolated from the internet.

Installation instructions

To set up a cluster with nodes that can host databases with multiple modules, you must follow this procedure on each node in the cluster:

1. [Install RS 5.5.](#)
2. To install the modules, run: `sudo ./install-modules.sh`
3. Either:
 - Set up the node as the [first node in the cluster](#).
 - [Join the node to an existing cluster](#).

Updated: March 7, 2023

Redis Enterprise Software Release Notes 5.4.14 (February 2020)

Redis Enterprise Software (RS) 5.4.14 is now available. This release bundles OSS Redis 5.0.7 and includes new Redis Modules versions, several enhancements, and bug fixes.

Overview

Follow these [instructions](#) for upgrading to RS 5.4.14 from RS 5.0.2 and above.

New features

- Version [5.0.7](#) of OSS Redis is merged into RS 5.4.14.
- The following GA releases of Redis Modules are bundled in RS 5.4.14:
 - [RedisBloom](#), version 2.2.1 (updated, [release notes](#))
 - [RedisGraph](#), version 2.0.1 (updated, [release notes](#))
 - [RedisJSON](#), version 1.0.4 (update, [release notes](#))
 - [RedisSearch](#), version 1.4.25 (updated, [release notes](#))
 - [RedisTimeSeries](#), version 1.2.3 (updated, [release notes](#))

Additional capabilities

- Added the ability to retrieve license details with a REST API command. Now you can get your license details from the admin console (settings > general) or from the REST API command:

```
GET https://localhost:9443/v1/license
```

The REST API response includes:
 - license - The cluster's name (FQDN) in the key string
 - expired - If the cluster key is expired (True or False)
 - activation_date - The date of the cluster key activation
 - expiration_date - The date of the cluster key expiration
 - shards_limit - The number of shards allowed by the cluster key
- Added Flush command in the UI console for Active-Active databases. The command flushes the data from all participating clusters.
- Updated `rladmin upgrade module` command so module arguments are optional. When you upgrade the module for a database, you can either:
 - Specify the module arguments to replace the existing arguments
 - Specify the `keep_module_args` flag to use the existing arguments

Examples:

1. To upgrade the version of RedisSearch to 10017 and replace the module arguments:

```
rladmin upgrade module db_name MyAwesomeDB module_name ft version 10017 module_args "PARTITIONS AUTO"
```

2. To upgrade RedisBloom to version 10100 and remove the current module arguments:

```
rladmin upgrade module db_name MyDB module_name bf version 10100 module_args " "
```

3. To upgrade RedisJSON to 10002 and use the current module arguments:

```
rladmin upgrade module db_name MyDB module_name ReJSON version 10002 keep_module_args
```

- Upgraded `js-yaml` version to 3.13.1, and `lodash` version to 4.17.15 (RS31819)

- Added an alert for actions that run for an extended period of time. For example, a notification is sent when an action like updating database property runs for more than a configurable threshold. Default value is set to 24 hours.

Information

- End of Life (EOL) for Redis Enterprise Software 5.4, as well as for Redis Modules and previous RS versions, can be found [here](#).
- Google Chrome browser on macOS Catalina requires self-signed certificate generated after June 2019 to include the extendedKeyUsage field in order to connect to the RS admin console. If you use a self-signed certificate that does not include this field, [update the self-signed certificate](#).
- When you upgrade an Active-Active Redis with active AOF from version RS 5.4.2 or lower to version RS 5.4.4 or higher:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.

Important fixes

- RS23396 - Improved the disconnection mechanism in case of inactive UI session
- RS27924 - Added descriptive error messages in the UI console when validating the license
- RS29968 - Improved internal mechanism to better support high scale of clients connections
- RS35675 - Updated the upgrade process of Active-Active Redis databases to save causal consistency and encryption flags
- RS36922 - Fixed an issue in a specific cluster upgrade scenario from versions earlier than RS 5.4.0 to RS 5.4.0 or higher

Known limitations

Upgrade

- When you upgrade a cluster from version 5.0.2-20 to version 5.4.14, you must first upgrade to version 5.2.2 and then to version 5.4.14.
- [RS 5.4.2](#) introduced new Active-Active Redis (CRDB) capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis (CRDB) is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the CRDB protocol](#).
- Before you upgrade a database with RediSearch Module to Redis 5.0, you must [upgrade the RediSearch Module](#) to version 1.4.2 or above.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, [contact Redis support](#).
- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the `keep_redis_version` option instead of `keep_current_version`.

Cluster API

- The API for removing a node is updated in RS 5.4.2 or higher. The API call must include json data and the “Content-Type: application/json” header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password
https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Discovery service

- For [Redis Sentinel \(Discovery Service\)](#), every database name must be unique across the cluster.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE command for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.4.10 (December 2019)

Redis Enterprise Software (RS) 5.4.10 is now available. This release includes an improved synchronization mechanism for Active-Active Redis and Replicas, several enhancements, and bug fixes.

Overview

Follow these [instructions](#) for upgrading to RS 5.4.10 from RS 5.0 and above.

New features

Synchronization mechanism in Active-Active Redis and Replica-of

RS 5.4.10 incorporates the improved [Redis synchronization mechanism \(PSYNC2\)](#) for Active-Active Redis (CRDB) and Replica-of.

As a result, failure scenarios in any A-A replica (and the source database of Replica-of), require only partial synchronization between the cross-region replicas instead of full synchronization that can be costly in time and bandwidth.

RS on RHEL 7 supports OpenSSL 1.0.2 and up

To keep RS secure and keep our internal libraries up-to-date, starting from RS 5.4.10 our RHEL 7 installations require a minimum of OpenSSL 1.0.2.

When you install or upgrade RS 5.4.10 on RHEL 7 with older version of OpenSSL, the installation fails with the error:

```
Error: Package: redislabs-5.4.10-1.rhel7.x86_64 (/redislabs-5.4.10-1.rhel7.x86_64)
      Requires: libcrypto.so.10(OPENSSL_1.0.2)(64bit)
```

If you see this error, upgrade to OpenSSL 1.0.2 or higher before you install RS.

Additional capabilities

- The following new GA releases of Redis Modules are bundled in RS 5.4.10:
 - RedisBloom, version 2.0.3
 - RedisSearch, version 1.4.18
 - RedisJson, version 1.0.4
 - RedisGraph, version 1.2.2
 - RedisTimeSeries, version 1.0.3
- Version 5.0.5 of Redis OSS, along with a fix to a corruption related to the HyperLogLog (that is part of [Redis 5.0.6](#)) are merged into RS 5.4.10.
- Using REST API, you can retrieve various license details such as activation date, expiration date, and the number of licensed shards. To get these details, run:

```
curl -v -u <user>:<password> https://localhost:9443/v1/license
```
- Updated PDNS version from 4.1.5 to 4.1.13.

Information

- End of Life (EOL) for Redis Enterprise Software 5.4, as well as for Redis Modules and previous RS versions, can be found [here](#).
- Google Chrome browser on macOS Catalina requires self-signed certificate generated after June 2019 to include the extendedKeyUsage field in order to connect to the RS admin console. If you use a self-signed certificate that does not include this field, [update the self-signed certificate](#).
- When you upgrade an Active-Active Redis with active AOF from version RS 5.4.2 or lower to version RS 5.4.4 or higher:
 - If replication is enabled, you must run the BGREWRITEAOF command on all replica shards after the upgrade.
 - If replication is not enabled, you must run the BGREWRITEAOF command on all shards after the upgrade.

Important fixes

- The titles of the 'rladmin status nodes' command output were updated from 'RAM' to 'FREE_RAM' (the amount of RAM in the node that is currently not used) and from 'AVAILABLE_RAM' to 'PROVISIONAL_RAM' (the amount of RAM in the node that can be provisioned).
- RS31492 - Upgraded dependent libraries: [python-cryptography to version 2.7](#); [NGINX to version 1.16.0](#); [PyYaml to version 5.1.2](#); [python-requests to version 2.22.0](#); [urllib3 to version 1.25.3](#)
- RS31187 - Upgraded the internal Python interpreter to version 2.7.16
- RS33042 - Fixed Support Package to contain complete SLOWLOG information
- RS32699 - Avoided unnecessary restart and failover of Redis processes when Active-Active database is upgraded
- RS32061 - Improved support of the Redis WAIT command
- RS31759 - Fixed failure during database import
- RS31747 - Fixed failure in upgrade from version 5.0.0-31 to 5.4.6-11
- RS30063 - Fixed the upgrade process when WatchdogAPI fails to bind to its port
- RS31477 - Fixed wrong calculation of node's 'AVAILABLE_RAM' ('PROVISIONAL_RAM') as displayed the output of 'rladmin status nodes' command

- RS30165 - Fixed failover scenario that did not take place during node restart
- RS29250 - REST API documentation was updated to include the SFTP and Mount Point backup/export options
- RS27327 - Improved the backup timing when using the database parameter of ‘backup_interval_offset’ through the REST API
- RS33883 - HCSAN command in Active-Active Redis updated to return Integer instead of a String.
- Fixed a limitation so Redis 5 and Redis 4 can be selected as the Redis version to use CRDB and RoF

Known limitations

Upgrade

- RS 5.4.2 introduced new Active-Active Redis (CRDB) capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis (CRDB) is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the CRDB protocol](#).
- Before you upgrade a database with RediSearch Module to Redis 5.0, you must [upgrade the RediSearch Module](#) to version 1.4.2 or above.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, [contact Redis support](#).
- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the `keep_redis_version` option instead of `keep_current_version`.

Cluster API

- The API for removing a node is updated in RS 5.4.2 or higher. The API call must include json data and the “Content-Type: application/json” header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password
https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Discovery service

- For [Redis Sentinel \(Discovery Service\)](#), every database name must be unique across the cluster.

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE command for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.4.6 (July 2019)

Redis Enterprise Software (RS) 5.4.6 is now available. This release includes the latest version of [Redis 5 \(5.0.5\)](#), bundles the GA release of the new [RedisTimeSeries](#) module, and adds other enhancements and bug fixes.

Overview

Follow these [instructions](#) for upgrading to RS 5.4.6 from RS 5.0 and above. If you have a version older than 5.0, you must first upgrade to version 5.2 (or at least 5.0).

New features

Time series

New GA release of the [RedisTimeSeries](#) (version v1.0.0) module is bundled in RS 5.4.6.

Modules versions

Updated versions of GA modules:

- [RedisTimeSeries](#) - v1.0.0 (new module)
- [RedisBloom](#) - v2.0.0 (version update)

- [RedisSearch](#) - v1.4.11 (version update)
- [RedisGraph](#) - v1.2.2 (no change)
- [RedisJSON](#) - v1.0.3 (no change)

Additional capabilities

- Latest version of Redis 5 (5.0.5) was merged into RS 5.4.6.
- If a user subscribes to a channel during recovery of a Active-Active Redis (CRDB) from an Append Only File (AOF), the user receives only the new messages. (Fixes known limitation from RS 5.4.4)
- CROSSLOT behavior now matches Redis OSS behavior for accessing multiple keys in a single command. Now, when you use CRC16 hash function (default), you can pass a command for different keys/tags that are mapped to the same slot. (RS23189)
- Cluster upgrade process was improved so that the duration of the cluster upgrade process reduced.

Information

- End of Life (EOL) for Redis Enterprise Software 5.4, as well as for Redis Modules and previous RS versions, can be found [here](#).

Important fixes

- RS28679 - Syncer and peers related statistics for Active-Active Redis (CRDB), which are already available with the REST API, are exposed through Prometheus.
- RS28946 - Metrics exporter related minor memory leak fix.
- RS26312 - Ports handling during installation fix.
- RS31703 - Fixed performance issue with MSET command.

Known limitations

Upgrade

- RS 5.4.2 introduced new Active-Active Redis (CRDB) capabilities that improve its compatibility with open source Redis. Now the string data-type in Active-Active Redis (CRDB) is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the CRDB protocol](#).
- Before you upgrade a database with RediSearch Module to Redis 5.0, you must [upgrade the RediSearch Module](#) to version 1.4.2 or above.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, [contact Redis support](#).
- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the `keep_redis_version` option instead of `keep_current_version`.

Cluster API

- The API for removing a node is updated in RS 5.4.2 or higher. The API call must include json data and the “Content-Type: application/json” header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password
https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4 and above
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE command for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.4.4 (June 2019)

Redis Enterprise Software (RS) 5.4.4 is now available. This release enables the functionality of Active-Active Redis (CRDB) combined with RoF (Auto Tiering), supports the creation of Redis databases with multiple modules, and adds other enhancements and bug fixes.

Overview

Follow these [instructions for upgrading to RS 5.4.4 from RS 5.0 and above](#). If you have a version older than 5.0, you must first upgrade to version 5.2 (or at least 5.0).

New features

Active-Active Redis with RoF

RS 5.4.4 lets you create Active-Active Redis databases (CRDBs) with Auto Tiering and get the benefits of geo-distributed Redis databases along with the significant cost savings of using Auto Tiering.

 Note: You must select **Redis 5** as the Redis version to use CRDB and RoF.

Support for multiple modules

RS 5.4.4 supports the creation and management of Redis databases with a combination of any of the following GA modules:

- RediSearch (GA)
- RedisGraph (GA)
- RedisBloom (GA)
- RedisJSON (GA)

To upgrade databases with multiple modules, you can use the `radmin upgrade` command.

 Note: The syntax for the `radmin upgrade` command now requires that the `module_args` parameter be written inside quotation marks.

- To upgrade only the modules within an existing database, for example with RedisBloom (bf) and RedisJSON:

```
radmin upgrade module db_name db1 module_name bf version 10003 module_args "" module_name ReJSON version 10001 module_args ""
```

- To upgrade a database with its modules, for example db1 with RediSearch and RedisGraph:

```
radmin upgrade db db1 and module module_name redisearch version 103 module_args "ON_TIMEOUT FAIL NOGC" and module module_name graph version 10016 module_args ""
```

You can also upgrade the modules with the REST API.

- To upgrade multiple modules, enter the details of each module in the `modules` parameter:

```
curl -X POST -u "demo@redislabs.com:password" -H "Content-Type: application/json" -d '{"modules": [{"module_name": "ReJSON", "current_module": "<module_uid>", "new_module": "<module_uid>", "new_module_args": "", "current_semantic_version": "1.0.4"}, {"module_name": "ft", "current_module": "<module_uid>", "new_module": "<module_uid>", "current_semantic_version": "1.4.3", "new_module_args": "PARTITIONS AUTO"}], "force_restart": true}' https://127.0.0.1:9443/v1/b dbs/2/modules/upgrade
```

- To upgrade a single module, you can either:

- Enter the details of the module in the `modules` parameter
- Enter the details of the module without the `modules` parameter, as in previous releases

Additional capabilities

- Support for persistence of AOF every 1 second in Active-Active Redis (CRDB), in addition to AOF every write. We recommend that you use AOF every 1 second for the best performance during the initial CRDB sync of a new replica.
- NGINX version updated to 1.14.2

Information

- Stay up to date with the [End of Life \(EOL\) Policy](#) for RS 5.4 and all previous RS versions.

Important fixes

- RS26508 - Fixed redis-cli failure for a specific CRDB data type.
- RS29191 - Fixed a failure when removing a replica from a CRDB.
- RS29097 - Fixed a misconfiguration when using SFTP backup and encounter a node failure.
- RS28286 - Updated SETEX and PSETEX commands output of CRDB to match Redis outputs.
- RS26984 - Fixed metrics_exporter reports for node and shard level metrics.
- RS19854 - Fixed uploading a Redis Module so you can upload a Redis Module when the admin console is connected to any node.
- RS29238 - Improved the compression performance in CRDB.

Known limitations

Upgrade

- RS 5.4.2 introduced new Active-Active Redis (CRDB) capabilities that improve its compatibility with open source Redis. Now the string data-type in CRDB is implicitly and dynamically typed, just like open source Redis. To use the new capabilities on nodes that are upgraded from version RS 5.4.2 or lower, you must [upgrade the CRDB protocol](#).
- Before you upgrade a database with RediSearch Module to Redis 5.0, you must [upgrade the RediSearch Module](#) to version 1.4.2 or above.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 or above by manually updating the certificates on the disk instead of [updating them through the API](#). For assistance with this issue, [contact Redis support](#).
- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- Starting from RS 5.4.2, to preserve the current Redis major.minor version during database upgrade you must use the `keep_redis_version` option instead of `keep_current_version`.

Subscriptions

- If a user subscribes to a channel during recovery of a CRDB from AOF, the user receives old messages that are stored in the AOF file.

Cluster API

- The API for removing a node is updated in RS 5.4.2 or higher. The API call must include json data and the "Content-Type: application/json" header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password  
https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Redis commands

- The capability of disabling specific Redis commands does not work on commands specific to Redis Modules.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4.x
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE command for string data-type in CRDBs return "string" (OSS Redis standard).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.4.2 (April 2019)

Redis Enterprise Software (RS) 5.4.2 is now available. This release improves the compatibility of Active-Active Redis (CRDB) with open source Redis, adds SFTP and Mount Points as backup destinations, email alerting and a number of other enhancements and bug fixes.

Overview

Follow these [instructions](#) for upgrading to RS 5.4.2 from RS 5.0 and above. If you have a version older than 5.0, you must first upgrade to version 5.2 (or at

least 5.0).

New features

Active-Active Redis (CRDB)

RS 5.4.2 introduces new Active-Active Redis (CRDB) capabilities which improve its compatibility with open source Redis. This simplifies the process of migration from open source Redis to Active-Active Redis:

1. The string data-type in CRDB is now implicitly and dynamically typed, just like open source Redis.
2. Geospatial data is now supported in CRDB.

These changes required upgrading the internal structures of CRDB and the communication protocol between the CRDB replicas. To take advantage of these updates with existing CRDBs you must [upgrade the CRDB protocol](#).

New backup destinations - SFTP and mount point

The storage options for [periodic database backup](#), [data export](#), and [data import](#) are extended to include:

- SFTP - a secured and well accepted protocol
- A local mount path that can point to network storage

These new options are in addition to the existing backup locations of FTP, AWS S3 and OpenStack Swift.

Email alerts per database (per team member)

Improved distribution of [email alerts](#) for cluster or database alerts. The cluster administrator can define, for each team member, the specific databases to receive alerts for, and whether to get the cluster alerts.

Optional client authentication (TLS)

You can now fine tune the [TLS configuration](#), and ease certificates management by excluding client authentication enforcement, so that the database clients, such as applications or other clusters, can connect to your database without authentication.

Node maintenance mode

When you do hardware or operating system maintenance on a server that hosts an Redis Enterprise node, it is important that you move all of the shards on that node to another node to protect the data. Starting with RS 5.4.2 you can use [maintenance mode](#) to handle this process simply and efficiently.

Additional capabilities

- Support for open source Redis version 5.0.4
- Viewer roles were further limited to improve security compliance. The DB Viewer and Cluster Viewer roles cannot view the Redis password of a database. The DB Viewer role also cannot view the 'log' screen.



Note: These improvements are also included in the REST API and are breaking-changes. If you use the relevant API requests with a DB Viewer or Cluster Viewer role, make sure that you update it accordingly.

- External user credentials for periodic backup destinations are now hidden. In REST API responses the password is hashed and in the admin console the password is displayed as asterisks.

Information

- For Redis Enterprise Software 5.4 End of Life (EOL), as well as for previous Redis Enterprise Software versions, see [product lifecycle](#).

Important fixes

- RS26985 - Fixed a failure of a DB with all-nodes policy, when a node is rebooted
- RS26433 - Fixed a scenario in which the CRDB sync stopped on one of the participating clusters
- RS25968 - Fixed a redis crash on RoF cluster
- RS25835 - Fixed a failure in periodic backup when AOF rewrite is in progress on the same shard
- RS25558 - Enabled updating absolute path in periodic backup FTP setup
- RS25381 - Fixed a race condition while updating a database parameter
- RS24188 - Fixed a failure during database upgrade
- RS23508 - Fixed a failure in 'rladmin status' command
- RS22485 - Fixed irregularities in service after increasing the size of a database which uses OSS cluster API

Known limitations

Upgrade

- Before you upgrade a database with RediSearch Module to Redis 5.0, the RediSearch [Module must be upgraded](#) to version 1.4.2 or above.
- Node upgrade fails if the SSL certificates were configured in version 5.0.2 and above by updating the certificates on the disk instead of using the new API. For assistance with this issue, see [SSL/TLS certificate updates](#) in our documentation or [contact Redis support](#).
- We recommend that you test module upgrade commands in a test environment before you upgrade modules in a production environment. The module upgrade arguments are not validated during the upgrade process and incorrect arguments can cause unexpected downtime.
- The rladmin option to use to preserve the current Redis major.minor version during database upgrade is now `keep_redis_version`, instead of `keep_current_version`.

Cluster API

- The API for node removal was updated. It must include json data, and “Content-Type: application/json” header. For example:

```
curl -X POST -H "Content-Type: application/json" -i -k -u user@redislabs.com:password  
https://localhost:9443/v1/nodes/3/actions/remove --data "{}"
```

Redis commands

- The capability of disabling specific Redis commands does not work on Redis Module specific commands.
- The CLIENT ID command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- CLIENT UNBLOCK command is not supported in RS 5.4 and RS 5.4.2
- Starting from RS 5.4.2 and after upgrading the CRDB, TYPE command for string data-type in CRDBs return “string” (OSS Redis standard).

Updated: August 31, 2022

Redis Enterprise Software Release Notes 5.4 (December 2018)

Redis Enterprise Software (RS) 5.4 is now available. RS 5.4 adds support for Redis 5.0 (GA) with the new Redis Streams data type.

Overview

You can upgrade to RS 5.4 from RS 5.0 and above according to the [upgrade instructions](#). If you have a version older than 5.0, you should first upgrade to version 5.2 (or at least 5.0).

New features

Redis 5.0 GA - Redis Streams

RS 5.4 adds support for Redis 5.0 (GA version- 5.0.2), which introduces the new [Redis Streams](#) data type. Redis Streams models a log data structure in-memory and implements additional powerful operations, such as Consumer Groups.

Redis graph module

Starting from RS 5.4, [Redis Graph](#), a new Redis Enterprise Module that introduces the world’s fastest graph database, is an integral part of Redis Enterprise package.

RedisGraph is the first queryable [Property Graph](#) database to use [sparse matrices](#) to represent the [adjacency matrix](#) in graphs and [linear algebra](#) to query the graph.

Active-Active Redis (CRDB) - Creation in non-clustering mode

In RS 5.4 you can [create Active-Active databases \(CRDBs\)](#) in a non-clustering mode. As a result, the following creation options are allowed:

1. Clustering mode - Creates a CRDB that consists of any number of shards in a clustering mode and is subject to [multi-key commands limitations](#).
2. Non-clustering mode - Creates a CRDB that consists of one shard only in a non-clustering mode so that [multi-key command limitations](#) do not apply.

High availability for replica shards

When [replica high availability](#) is enabled and a master shard fails, a replica (formerly *slave*) shard is automatically promoted to a master shard to maintain data availability. This creates a single point of failure until a new replica shard is manually created.

RS 5.4 expands the high availability capabilities by adding the ability to automatically avoid this single point of failure by configuring the cluster to automatically migrate the replica shard to another available node. In practice, replica migration creates a new replica shard and replicates the data from the master shard to the new replica shard.

**Note that just as is the case with the Redis open-source project, Redis is in the process of changing the “master-replica” terminology to “master-replica” everywhere, including within our documentation.*

Additional capabilities

- Support for new operating systems- Ubuntu 18.04 and RHEL 7.6.

Product version lifecycle

- The End of Life (EOL) for Redis Enterprise Software 4.5.X was November 30th, 2018, in accordance with our [published policy](#). We recommend that customers with version 4.5 or below upgrade to the latest version.

Important fixes

- RS23616 - Fixed a failure when updating the memory limit of RoF database.
- RS22871 - Fixed a certificate verify failure after nodes upgrade.
- RS2862 - Improved admin console performance in case multiple browsers or windows are directed to the admin console.
- RS22751 - Fixed an issue in the backup process which caused temporary service outage.
- RS22636 - Fixed Redis process failure when a ReJSON Module's command is executed.
- RS22601 - Fixed a failure during shard migration procedure.
- RS22478 - Fixed a failure in replica-of process between two databases with ReBloom Module.
- RS21974 - SMTP username and password are not mandatory in the email server settings when there is no need for authentication.
- RS21801 - Fixed admin console issues when cluster is configured with FIPS compliance.
- RS21772 - Fixed a failure when trying to update a database's endpoint policy to all-master-shards.
- RS19842 - Updated permissions of some internal files.
- RS19433 - Improved RAM eviction process for RoF databases.
- RS18875 - Added the ability to upgrade database gradually, few shards at a time.
- RS15207 - Fixed a failure during re-shard operation.

Known limitations

Installation

- In default Ubuntu 18.04 installations, port 53 is in use by systemd-resolved (DNS server). In such a case, the system configuration must be changed to make this port available before running RS installation.

Upgrade

- Before you upgrade a database with the RediSearch module to Redis 5.0, you must [upgrade the RediSearch module on that DB](#) to 1.4.2 or higher. We recommend that you upgrade the RediSearch module before you upgrade the cluster to RS 5.4.
- Node upgrade fails if SSL certificates were configured in version 5.0.2 and above by updating the certificates on the disk instead of using the new API. For assistance with this issue, please [contact Redis support](#).

Cluster API

- Removed the deprecated argument `backup_path` from cluster API. To create or update BDBs, please use `backup_location`.

Redis commands

- The capability of disabling specific Redis commands does not work on Redis Module specific commands.
- The `CLIENT ID` command cannot guarantee incremental IDs between clients that connect to different nodes under multi proxy policies.
- The length of the `socket_path` variable, which is defined in a node, cannot exceed 88 characters.
- `CLIENT UNBLOCK` command is not supported in RS 5.4.

Redis Enterprise Software 5.2.2 (August 2018)

Redis Enterprise Software (RS) 5.2.2 is now available.

RS 5.2.2 is a minor version that includes important fixes and minor enhancements to Redis Enterprise 5.2.

Overview

If you are upgrading from a previous version, make sure to review the upgrade instructions before beginning the upgrade process. You can upgrade to RS 5.2.2 from RS 4.5 and above. If you have a version older than 4.5, you should first upgrade to version 5.0 (or at least 4.5).

New capabilities

Support for Redis version 4.0.10 RediSearch Enterprise, which is installed with Redis Enterprise Software by default, has been updated to a newer version (1.4.0)

Important fixes

- RED-21080 – Fixed the memory limit calculation for RoF databases
- RED-20825 – Updated the ‘RAM limit’ of RoF databases to a range between 10% and 100%
- RED-20506 – Fixed high CPU and File Descriptors utilization by the node watchdog process (node_wd)
- RED-20275 – Fixed wrong metrics values while importing a dataset into an RoF database
- RED-20214 – Fixed obstacles to login to the UI when using an LDAP integration
- RED-20162 – Fixed known limitation of being able to activate “Require SSL for All Communication” to Redis Enterprise CRDBs via Rest API without providing a certificate
- RED-19758 – Upgraded NGINX from 1.10.3 to 1.13.12
- RED-19415, RED-18945 – Improved support for Lettuce client with OSS Cluster enabled
- RED-19287 – Fixed a scenario of a stuck shard migration process
- RED-18459 – Updated the data persistence (AOF / Snapshot) of RoF databases to be handled, by default, by the replica shard/s
- RED-20541 – Improved handling of aof file when its tail is corrupted (using aof-load-corrupt-tail flag)
- RED-21936 – Improved handling of CRDB configuration update when URL parameter was supplied at creation time
- RED-19760 – Added the capability to control the minimum TLS version that can be used for encrypting the Discovery Service data

Known limitations

- When updating the general settings of a cluster, the ‘username’ and ‘password’ fields in the email server settings cannot be left empty. In case one wants to update the general settings and prefer to leave the ‘username’ and ‘password’ fields empty, the REST API should be used.
- An issue prevents the user from defining ‘min_data_TLS_version’ on the source cluster when working with Replica Of or CRDB.

Updated: August 31, 2022

Redis Enterprise Software Release Notes 5.3 BETA (July 2018)

Redis Enterprise Software (RS) 5.3 is now available.

RS 5.3 is a preview version that includes all the capabilities of Redis Enterprise 5.2, plus support for Redis 5.0 with the new data type, which is called Streams.

New features

RS 5.3 adds support for Redis 5.0 and is based on its latest Release Candidate (RC3). Redis 5.0 exposes the new Redis Streams data type, which provides a super fast in-memory abstraction of an append-only log. For more information and usage examples, check out the Streams documentation [here](#).

Preview considerations

As a preview version, we do not support upgrading to RS 5.3 from any previous version or upgrading from RS 5.3 to any future version. In addition, we do not commit to fixing bugs in RS 5.3.

Known limitations

The RS 5.3 preview version does not support Enterprise Modules (- RediSearch, ReJSON and ReBloom), Auto Tiering or active-active Redis (CRDB).

Updated: August 17, 2023

Redis Enterprise Software Release Notes 5.2 (June 2018)

Redis Enterprise Software (RS) 5.2 is now available. Key features include new data types and causal consistency for active-active (also known as the Redis CRDT or CRDB Conflict-free Replicated Database), as well as enhanced security features.

Overview

If you are upgrading from a previous version, make sure to review the upgrade instructions before beginning the upgrade process. You can upgrade to RS 5.2 from RS 4.5 and above. If you have a version older than 4.5, you should first upgrade to version 5.0 (or at least 4.5).

New features

Active-Active Redis (CRDB) supports sorted sets and lists

RS 5.2 adds active-active Redis (CRDB) support for Sorted Sets and Lists. Now all major Redis data types are supported with CRDT, so you can use Redis Enterprise in an active-active manner for all your Redis use cases, with seamless conflict resolution. [Click here](#) for more information about how to develop applications with geo-replicated CRDBs.

Causal consistency in Active-Active Redis (CRDB)

Causal consistency in active-active Redis (CRDB) guarantees that the order of operations on a specific key will be maintained across all CRDB instances. For instance, if operations A and B were applied on the same key, and B was performed after the effect of A was observed by the CRDB instance that initiated B, then all CRDB instances would observe the effect of A before observing the effect of B.

This way, any causal relationship between operations on the same key is also observed and maintained by every replica. Such capability is important for various applications, e.g. social network status updates or chat applications (assuring that the chronology of messages doesn't get mixed up). For more information, [click here](#).

Enhanced security features

- Admin action audit trail

Management actions performed with Redis Software are audited in order to fulfill two major objectives: (1) make sure that system management tasks are appropriately performed and/or monitored by the Administrator(s), and (2) facilitate compliance with regulatory standards such as HIPAA, SOC2, PCI, etc.

In order to fulfill both objectives, audit records contain the following information: (1) who performed the action, (2) what exactly was the performed action, (3) when the action was performed, and (4) whether the action succeeded.

To get the list of audit records/events, one can use the REST API or the Log page in the UI; the Log page displays the system and user events regarding alerts, notifications and configurations.

- Ability to disable TLS versions

Version 5.2 introduces the option to set the minimum TLS version that can be used for encrypting the various flows. You can do so using the REST API or the following rladmin commands:

- For the management UI and the REST API:

```
rladmin> cluster config min_control_TLS_version [version, e.g. 1.2]
```

- For the data path encryption:

```
rladmin> cluster config min_data_TLS_version [version, e.g. 1.2]
```

Note that communications that use older TLS versions will not be allowed.

- HTTPS enforcement

With this feature, you can restrict REST API access to only those using HTTPS. In order to do so, use the REST API or the following rladmin command:

```
rladmin> cluster config http_support [disabled | enabled]
```

Additional capabilities

- Support for Redis version 4.0.9
- Fixes for Redis important security issues related to the Lua scripting engine

Information

- The end-of-service-life (EOSL) for Redis Enterprise Software 4.4.X is June 30th, 2018, in accordance with our [published policy](#). We recommend that customers running version 4.4 or below upgrade to the latest version.

Important fixes

- RS19755 - Added the ability to enforce HTTPS communication when accessing REST API
- RS19571 - Fixed RHEL installation when FIPS mode is enabled
- RS19490 - Fixed a failure while upgrading from 4.5 to 5.0.2
- RS19475 - Added the ability to disable TLS versions for the control and data paths
- RS18712 - Fixed a failure in endpoint failover
- RS17208 - Added an option to disable Redis commands via Rest API
- RS14973 - Updated NGINX.conf template to turn off exposing the server version
- RS11181 - Set file permissions of log files, redis-conf files, rdb files and their rotations to "640"

Known limitations

- After creating Redis Enterprise CRDBs which “Require SSL for CRDB communication only,” one can set definitions via Rest API to activate “Require SSL for All Communication” without providing a certificate. In such case, connections to the cluster will be blocked.

Updated: August 31, 2022

Redis Enterprise Software 5.0.2 (2018 March)

Redis Enterprise Software 5.0.2 is now available. Key features include functional and performance updates for CRDB, changes to module deployment, and general fixes.

Overview

If you are upgrading from a previous version, make sure to review the upgrade instructions before beginning the upgrade process.

You can upgrade to RS 5.0.2 from RS 4.4.2 and above. If you have a version older than 4.4.2, you must first upgrade to at least 5.0.



Note: Starting from RS 5.0.2, ports 3338 and 3339 should also be opened on each node for the purpose of internal cluster communication. For more information, check the '[network port configurations](#)' page

New features

CRDBs

- The ability to add and remove participating clusters from a CRDB
- Communications between participating clusters can be encrypted using SSL/TLS
- Imports can be done to an existing database without flushing the existing data beforehand.

Modules

- Redis Enterprise Modules are installed with Redis Enterprise Software by default
- RedisBloom and RediSearch Enterprise have been updated to newer versions

Other

- Discovery Service supports encryption using SSL/TLS
- Starting from version 5.0.2 build #30, Redis Enterprise Software is supported on RHEL 7.5

Important fixes

- RS16153 – Supervisord version update
- RS16667 - Fixed issue with 'rladmin status' timeout
- RS17746 - Fixed upgrade issue when -s option used
- RS17997, RS18088 - Upgrade issues when using non-default socket file path
- RS8584 – Endpoint migration provided misleading plan message
- RS17696 - Fixed issues with Multi-proxy and intermittent network issues
- RS17362 - Upgrade fails under some circumstances
- RS18351 - Listener active after node has been declared dead
- RS18874 - Fixed OOM issue due to redis_mgr high memory consumption
- RS19002 - Fixed wrong message when an upgrade of a quorum node with all-nodes policy takes place

Important Fixes in Build #30:

- RS19701 - Fixed high CPU usage on large scale clusters
- RS19869- Added support for Redis version 4.0.9
- RS20153- Fixed Redis important security issues related to the Lua scripting engine
- RS19852- Fixed proxy crash which might happen for SSL-enabled DBs

Known limitations

- Since Redis Enterprise CRDBs have counters, unlike traditional Redis databases, they must be handled differently when importing. There is a special type of import because of importing counter data types. When performing the import through the admin console, you will be prompted to confirm you want to add the data to the CRDB or stop and go flush the database.
- This version of RS comes with a pre-bundled python which might over-ride your default installed python version, this can be solved by changing your PATH environment variable.
- Uploading a Redis Module through the admin console, can be performed only when the admin console is connected to the master node.
- Write operations are not allowed for database which was created with password of exactly 50-characters.

Updated: June 22, 2022

Redis Enterprise Pack 5.0 Release Notes (November 2017)

Redis Enterprise Software 5.0 is now available. Key features include Geo-Distributed Active-Active Conflict-free Replicated Databases (CRDB), LDAP integration, Redis Module integration, and support for the Redis Cluster API.

Overview

If you are upgrading from a previous version, make sure to review the upgrade instructions before beginning the upgrade process.

You can upgrade to RS 5.0 from RS 4.4.2 and above. If you have a version older than this, you must first upgrade to at least 4.4.2.

New features

Support for Redis Cluster API

The Redis Cluster API support in Redis Enterprise Software (RS) provides a simple mechanism for Cluster enabled Redis clients to learn and know the cluster topology. This enables clients to connect directly to an RS proxy on the node hosting the master shard for the data being operated on. The result is that for all but the initial call to get the cluster topology or reacquire the location of the master shard, the client will connect to the RS endpoint proxy where the master shard is located. [Learn more about the Cluster API implementation](#).

Geo-distributed Active-Active conflict-free replicated databases (CRDB)

Developing globally distributed applications can be challenging, as developers have to think about race conditions and complex combinations of events under geo-failovers and cross-region write conflicts. CRDBs simplify developing such applications by directly using built-in smarts for handling conflicting writes based on the data type in use. Instead of depending on just simplistic “last-writer-wins” type conflict resolution, geo-distributed CRDBs combines techniques defined in CRDT (conflict-free replicated data types) research with Redis types to provide smart and automatic conflict resolution based on the data type’s intent.

For more information, go here. For information, go to [Developing with CRDBs](#).

Redis modules

Redis Modules enable you to extend the functionality of Redis Enterprise Pack. One can add new data types, capabilities, etc. to tailor the cluster to a specific use case or need. Once installed, modules benefit from the high performance, scalability, and high availability that Redis Enterprise is known for.

Redis modules

Redis has developed and certified the following modules for use with Redis Enterprise Pack:

- [RediSearch](#)
 - This module turns RS into a supercharged distributed in-memory full-text indexing and search beast.
- [ReJSON](#)
 - Now you have the convenience JSON as a built-in data type and easily able to address nested data via a path.
- [RedisBloom](#)
 - Enables RS to have a scalable bloom filter as a data type. Bloom filters are probabilistic data structures that quickly determine if values are in a set.

Custom modules

In addition, Redis Enterprise Pack provides the ability to load and use custom [Redis modules](#) or of your own creation.

Support for Docker

Deploying and running your Redis Enterprise Software cluster on Docker containers is supported in development systems and available to pull from Docker hub. With the official image, you can easily and quickly test several containers to build the scalable and highly available cluster Redis Enterprise Software is famous for.

For more information, go to [quick start with Redis Enterprise on Docker](#).

LDAP Integration

As part of our continued emphasis on security, administrative user accounts in Redis Enterprise Pack can now use either built-in authentication or authenticate externally via LDAP with saslauthd. The accounts can be used for administering resources on the cluster via command line, Rest API, or admin console.

For more information see [LDAP Integration](#).

Additional capabilities

Support for additional Redis commands and features:

- Support for Redis version 4.0.2
- Support added for RHEL/CentOS 6.9 and 7.4

Information

- In the node bootstrap API, the structure of the JSON has changed for adding an external IP during the bootstrap process.
- End-of-Life for RHEL/CentOS 6.5 and 6.6 have been reached, so those versions are no longer supported.
- Modules are not supported in Redis Enterprise Pack 5.0 for RHEL/CentOS 6.x

Important fixes

5.0.0-31

- RP9299 - Issue with reliability of metric ingress
- RP9680 - Redis Enterprise Pack starting before /etc/rc.local script executed

- RP12363 - In some cases, flash drives do not mount following a stop/start of the node
- RP12493 - Allow change to debug package creation location
- RP13079 - DNS doesn't change after having removed the external IP address in some cases
- RP13933 - rladmin balance sometimes shows incorrect information
- RP14060 - pubsub stats aren't reflected correctly by the stats archiver
- RP14939 - Add license checks to all needed entry points in the cluster
- RP15090 - Problem in log rotate script causes other logs to not rotate
- RP15104 - rlutil check fix doesn't work sometimes
- RP15130 - Fixed permission on two logs with incorrect ownership
- RP15160 - Allow option to change ports for API and CM in NGINX
- RP15164 - Allow unix socket folder to be configurable at build time
- RP15499 - rladmin command not responding on a cluster with large number of shards
- RP15853 - When trying to add a new db the 'activate' button was changed to 'update' and was grayed out.
- RP15861 - Allow unix socket folder to be configurable at install time
- RP16115 - TTL bug with Replica Of and import
- RP16447 - DMC client connection reports incorrect number of connections to monitoring applications.
- RP16481 - In some cases, resource_mngr uses incorrect directories to compute persistent and ephemeral storage

Updated: August 17, 2023

Redis Enterprise Pack 4.5 Release Notes (May 2017)

If you are upgrading from a previous version, make sure to review the upgrade instructions before beginning the upgrade process.

You can upgrade to this version from any 4.4 version. If you have a version older than 4.4 you must first upgrade to 4.4 or higher, and only then upgrade to this version.

New features

The new discovery service with support for Redis Sentinel API

The Discovery Service provides an IP-based connection management service used when connecting to Redis Enterprise Pack databases. When used in conjunction with Redis Enterprise Pack's other high availability features, the Discovery Service assists an application cope with topology changes such as adding, removing of nodes, node failovers and so on. It does this by providing your application with the ability to easily discover which node hosts the database endpoint. The API used for discovery service is compliant with the Redis Sentinel API.

Discovery Service is an alternative for applications that do not want to depend on DNS name resolution for their connectivity. Discovery Service and DNS based connectivity are not mutually exclusive. They can be used side by side in a given cluster where some clients can use Discovery Service based connection while others can use DNS name resolution when connecting to databases.

Building large databases with RAM and Flash memory in Auto Tiering v2.0

With Redis Enterprise Pack 4.5, Auto Tiering v2 is production ready. RFv2 brings performance, reliability, and stability enhancements as well as many features customers have been waiting for.

Auto Tiering offers users of Redis Enterprise Pack and Redis Enterprise Cloud Private the unique ability to operate a Redis database that spans both RAM and flash memory (SSD), but remains separate from Redis Enterprise Pack's persistence mechanisms. Whilst keys are always stored in RAM, RoF intelligently manages the location of their values (RAM vs Flash) in the database via a LRU-based (least-recently-used) mechanism. Hot values will be in RAM and infrequently used, while warm values will be ejected to flash memory. This enables you to have much larger datasets with RAM-like latency and performance, but at dramatically lower cost than an all-RAM database.

Additional capabilities

Support for additional Redis commands and features:

- The Redis TOUCH command is now supported
- Redis version upgraded to 3.2.8
- Support for OBJECT in Lua scripts

Support has been added RHEL 7.3 with this version.

Important fixes

- RP10106 - SSL Certificate should not need to be signed with a stronger hashing algorithm to be accepted
- RP10465 - failover times can be higher under certain scenarios in local watchdog profile
- RP10633 - Improve install.sh and answers file

- RP11880 - Improved replica sync and add node robustness
- RP8689 - Minimized impact when changing RAM-Flash limit on Redis Enterprise Flash
- RP12063 - Improved Redis Flash data import/population performance
- RP11608 - Improvements to databases.txt creation in support package
- RP11941 - Eliminated warning and errors during upgrade on RHEL6 with leash and python2.6 is installed
- RP11994 - Databases under certain cases may not display in UI even though they are in the cluster metadata and safely operating.
- RP12438 - Email alerts with Amazon SES may fail under certain conditions.
- RP12538 - Redis failover was initiated by node_wd during sync to new replica
- RP10264 - Improved debuginfo package for better supportability

Important fixes in RP 4.5.0-22

- RP12667 - NGINX security improvements, improved TLS and SSL related warnings on security scans.
- RP12690 - Added simpler ability to recover cluster from AWS S3
- RP13359 - Advanced memory allocation and booking enhancements

Important fixes in RP 4.5.0-31

- RP13060 - Client may experience reconnect issues after failover of the endpoint.
- RP13711 - Disabling IPv6 may cause startup of node services to fail (NGINX) on RHEL 7
- RP12747 - Calculation can be incorrect for memory quota for databases

Important fixes in RP 4.5.0-35

- RP12844 - cnm_exec crashes on “not enough arguments for format string”

Important fixes in RP 4.5.0-43

- RP9846 - In UI, Used Memory may show incorrect values
- RP12211 - UI fails import from a Redis OSS DB with password
- RP13356 - Enable failover when license expires
- RP14692 - radmin status command may crash during backup
- RP14541 - In rare cases, DMC log grew quickly and caused stability issues
- RP15107 - When using Auto Tiering, it may cause DMC proxy crashes

Important fixes in RP 4.5.0-47

- Multiple important Auto Tiering updates.

Important fixes in RP 4.5.0-51

- 15161 - Make Unix socket folder configurable at install time
- 15164 - Make Unix socket folder configurable at build time
- 16082 - Make Unix socket folder configurable at runtime
- Fixed an issue where on import of data, TTL information was set incorrectly

Updated: August 17, 2023

RLEC 4.4 Release Notes (December 2016)

If you are upgrading from a previous version, make sure to review the [upgrade instructions](#) before beginning the upgrade process.

You can upgrade to this version from any 4.3 version. If you have a version older than 4.3 you must first upgrade to 4.3 and only then upgrade to this version.

New features

- Databases can now be configured to have multiple proxies for improved performance. Note that when you upgrade the cluster to this version and then upgrade existing databases, the databases will be updated to use the Single proxy policy and Dense shard placement policy. For additional details, refer to [Multiple active proxies](#).
- Support for Redis version 3.2 added. When you install or upgrade the cluster the new default version for Redis databases will be 3.2 and when you upgrade the databases they will be updated to this version. If you would like to change the default version to Redis 3.0, refer to the instruction in the [Upgrading databases](#). If you would like to upgrade existing databases to the latest 3.0 minor version, refer to the Known Issues section below.
- The cluster can now be configured to support both private and public IPs to connect to database endpoints through both public and private networks. For additional details, refer to [Private and Public Endpoints](#).
- **radmin status** command output has been enhanced to include an indication on which node radmin is running by adding the “*” sign next to the node entry, and to show the host name of the machine the node is running on.

- Users can now be assigned security roles to control what level of the databases or cluster the users can view and/or edit.

Changes

- As result of adding the support for multiple proxies for a database, the following changes have been made:
 - When you upgrade the cluster to this version and then upgrade existing databases, the databases will be updated to use the Single proxy policy and Dense shard placement policy.
 - **rladmin status** command output has been updated.
 - **failover [db <db:id | name>] endpoint <id1 .. idN>** and **migrate [db <db:id | name> | node <origin node:id>] endpoint <id> target_node <id>** commands are no longer relevant for databases using the **single | all-master-shards | all-nodes** proxy policy. Instead proxies can be bound or unbounded to databases as needed.
 - New **rladmin** commands were added, such as **bind** and **placement**.
- RLEC has been updated to remove the need to use **sudo** in runtime. You still need to be root or use **sudo** when initially installing RLEC.
- You no longer need to be root or use **sudo** to run the **rladmin** command, now it is preferred to be a non-privileged user that is member of the **redislabs** group to run the command.
- All cluster services are now run using the supervisor mechanism. As a result starting, stopping and restarting RLEC services should be done using **supervisorctl** command from the OS CLI.
- Linux OS **vm.swappiness** is now advised to be set to zero, for more information see [Disabling Swap in Linux](#).

Important fixed issues since 4.3.0

- RLEC-7542 - Add ability to create and manage role based user security
- RLEC-8283 - The cluster recovery process does not work properly when the cluster that needs to be recovered does not have a node with ID 1.
- RLEC-8284 - Add functionality to **rladmin** to mark a node as a quorum only node
- RLEC-8498 - Backup fails under rare conditions
- RLEC-8579 - **rladmin** supports uppercase for **external_addr** value
- RLEC-8656 - Fixed conflict with SELinux
- RLEC-8687 - Fixed issue where strong password requirements were not honored correctly.
- RLEC-8694 - DMC failed while creating DB with 75 (150 replicated) shards
- RLEC-8700 - Fixed issue with network split scenario
- RLEC-8833 - Fixed issue where in some cases endpoint were not getting new IPs after node replacement.
- RLEC-9069 - Fixed issue related to RHEL 7 and IPv6.
- RLEC-9156 - Fixed issue causing a full resync of data when a source or destination failure occurred.
- RLEC-9173 - Issue with writing data after master and replica failed
- RLEC-9235 - Issue with SSL connection error and self signed certificates
- RLEC-9491 - Fixed alerting issue due to incorrect measurement
- RLEC-9534 - Fixed issue with node remove command after RLEC uninstalled
- RLEC-9658 - Failed to import backup file from FTP server.
- RLEC-9737 - Fixed issue with backup process to use ephemeral storage when needed
- RLEC-9761 - UI had incorrect value increments
- RLEC-9827 - Server with a high number of cores and running RHEL can have issues running **systune.sh**
- RLEC-9853 - Fixed issues with logrotate on RHEL 7.1 so it runs as non-privileged user
- RLEC-9858 - If proxy crashed, in some cases this would prevent completion of redis failover process
- RLEC-9893 - DB recovery process doesn't recognize original rack name when in uppercase
- RLEC-9905 - x.509 certificate signed by custom CA cannot be loaded in UI
- RLEC-9925 - master endpoint and shards goes down if co-hosted with master of the cluster and the node goes down (single proxy policy)
- RLEC-9926 - Master shard could remain down if on the same node as the master of the cluster and the entire node goes down
- RLEC-10340 - Fixed a typo that crashed **rladmin status** output in some cases

Changes in 4.4.2-42:

- RLEC-11941 - Upgrade to 4.4.2-35 on RHEL6 - leash failed when python2.6 is installed
- RLEC-11994 - RLEC 4.4.2-35: the UI doesn't display the DBs with replication

Changes in 4.4.2 - 49

- RLEC-11209 - Unable to run upgrade due to **running_actions** check
- RLEC-12647 - Backup to S3 with periods in bucket name are failing in some cases

Known issues

- **Issue:** When upgrading to this version from a previous RLEC version, **rladmin status** output will show the database status as having an old version. When you upgrade the Redis database (using **rladmin upgrade db** command) the Redis version will be updated to 3.2 even if you updated the cluster's Redis default version to 3.0. **Workaround:** If you would like to cancel the old version indication in **rladmin status** without upgrading the Redis version to 3.2 you should run the **rladmin upgrade db** command with the **keep_current_version** flag which will ensure the database is upgraded to the latest 3.0 version supported by RLEC.
- **Issue:** RLEC-9200 - in a database configured with multiple proxies, if a client sends the MONITOR, CLIENT LIST or CLIENT KILL commands, only commands from clients connected from the same proxy are returned instead of all commands from all connections. **Workaround:** If you would like to

get a result across all clients, you need to send the monitor command to all proxies and aggregate them.

- **Issue:** RLEC-9296 - Different actions in the cluster, like node failure or taking a node offline, might cause the Proxy policy to change Manual.
Workaround: You can use the `rladmin bind [db <db:id | name>] endpoint <id> policy <single | all-master-shards | all-nodes>` command to set the policy back to the required policy, which will ensure all needed proxies are bounded. Note that existing client connections might disconnected as result of this process.
- **Issue:** RLEC-8787 - In some cases when using the replica-of feature, if the source database(s) are larger than the target database, the memory limit on the target database is not enforced and that used memory of the target database can go over the memory limit set. **Workaround:** You should make sure that the total memory limit of all source databases is not bigger than the memory limit of the target database.
- **Issue:** RLEC-8487 - Some Redis processes stay running after purging RLEC from the machine and causes an attempt to reinstall RLEC to fail.
Workaround: Run the purge process for a second time and ensure that the Redis processes were removed.
- **Issue:** RLEC-8747 - When upgrading to this version, if the UI is open in the browser the UI might not work properly after the upgrade. **Workaround:** Refresh the browser and the UI will return to work properly.
- **Issue:** In the Replica Of process, if the target database does not have replication enabled and it is restarted or fails for any reason, the data on the target database might not be in sync with the source database, although the status of the Replica Of process indicates that it is. **Workaround:** You must manually stop and restart the synchronization process in order to ensure the databases are in sync.
- **Issue:** In the Replica Of process, if the source database is resharded while the Replica Of process is active, the synchronization process will fail.
Workaround: You must manually stop and restart the synchronization process after the resharding of the source database is done.
- **Issue:** In the Replica Of process, if there is very high traffic on the database the Replica Of process might be restarted frequently due to the "replica buffer" being exceeded. In this case, you will often see the status of the Replica Of process display as "Syncing". **Workaround:** You must manually increase the "replica buffer" size through radmin. To find the appropriate buffer size please contact support at: support@redislabs.com.
- **Issue:** In a cluster that is configured to support rack-zone awareness, if the user forces migration of a master or replica shard through radmin to a node on the same rack-zone as its corresponding master or replica shard, and later runs the rebalance process, the rebalance process will not migrate the shards to ensure rack-zone awareness compliance. **Workaround:** In the scenario described above, you must use radmin to manually migrate the shard to a node on a valid rack-zone in order to ensure rack-zone awareness compliance.
- **Issue:** DNS doesn't change after having removed the external IP address. **Workaround:** Unbind IP from affected node and then bind it back.
- **Issue:** CCS gets an error and won't start if /var/opt/redislabs/persist/ does not exist. **Workaround:** Make sure this directory is not deleted and continues to exist.

Updated: August 17, 2023

RLEC 4.3.0-230 Release Notes (August 2, 2016)

If you are upgrading from a previous version, make sure to review the [upgrade instructions](#) before running through the upgrade process.

You can upgrade to this version from any 4.2 version. If you have a version older than 4.2 you should first upgrade to 4.2 and only then upgrade to this version.

New features

- Various improvements to internal performance and stability were implemented.
- RLEC Flash functionality added. For additional details, refer to [Auto Tiering](#) and contact support@redislabs.com if you are interested in this functionality.
- Support for Redis version 3.0 added. When you install or upgrade the cluster the new default version for Redis databases will be 3.0 and when you upgrade the databases they will be updated to this version. If you would like to change the default version to Redis 2.8 refer to the instruction in the [Upgrading databases](#) section. If you would like to upgrade existing databases to the latest 2.8 minor version, refer to the Known Issues section below.
- Complete cluster failure recovery instructions added. For additional details, refer to [Cluster Recovery](#).
- Major improvements made to database replication performance process by using diskless replication between master and replica shards. The data between the master and replica shards is streamed directly, instead of using the default file-on-disk mechanism. This behavior can be changed for the entire cluster or per database through radmin.
- Major enhancements made to radmin command line interface to add new administration functionalities.
- rlcheck installation verification utility added to facilitate checking node health. For additional details, refer to [rlcheck installation verification utility](#).
- Added the ability to allow the user to configure how machine IP addresses are used in Node Configuration setup in the management UI. For additional details, refer to [Initial setup - creating a new cluster](#).
- Connection to database endpoint can now be encrypted with SSL. For additional details, refer to [Securing client connection with SSL](#).
- Added support for running the cluster on the following operating systems and versions: RHEL/CentOS 6.6, 7.1, 7.2, RHEL 6.7, Oracle Linux 6.5.

Changes

- Environment configuration profile with name "default" has been changed to "cloud" and the default value has been changed to "local-network". For additional details, refer to [Performance optimization](#) section.
- In the REST API, when creating a database and not setting the database replication parameter to "true", the default value assigned by the cluster has changed from "true" to "false".
- radmin syntax updates can affect commands written for prior versions of RLEC. In this version commands that are run directly from the operating system CLI prompt (not through the radmin prompt) no longer require quotation marks for text with special characters.
- Option added to the Replica-of process that allows gradual "shard-by-shard" replication of a sharded database, reducing the load on internal buffers. This optimization setting can be configured on the target database using the `gradual_sync_mode` parameter in radmin.
- The functionality for taking a node offline was removed from the UI.

Fixed issues

- RLEC-7110 - node does not recover properly after restart in case ephemeral storage is not available yet
- RLEC-7502 - log rotate job not working properly on RHEL operating system
- RLEC-7599 - issues running on a server with no IPv6 kernel support
- RLEC-7561, RLEC-7597 - issues connecting to database endpoint as result of cluster name containing capital letters
- RLEC-7245 - on machines with multiple IPs sometimes the wrong IP address is chosen for internal traffic
- RLEC-6815 - wrong log entry is added when enabling cluster alert regarding database version compatibility
- RLEC-7652 - database is down in certain failover scenarios only when the database is completely empty
- RLEC-7737 - issue where in a specific scenario after node restarts, a database with replication both master and replica shards are reported as down
- RLEC-7712 - in some cases, the Replica Of process may fail when Redis password is set
- RLEC-7726 - node object "avg_latency" statistic is not returned in the REST API
- RLEC-7358 - install script issue when running on LVM disks
- RLEC-8086 - port 9443 missing from redislabs-clients.xml
- RLEC-7281 - rotation of internal log files not working properly
- RLEC-8279 - updates to a user definition might cause password reset to be required
- RLEC-8512 - when upgrading an existing cluster that has uppercase letters in the cluster name (FQDN) the cluster might not function properly after the upgrade and attempts to connect to a database might fail
- RLEC-8371 - email alerts do not work when using Amazon SES service
- In certain scenarios the node upgrade process may fail if the node is in the offline state

Known issues

- **Issue:** When upgrading to this version from a previous RLEC version, rladm status output will show the database status as having an old version. When you upgrade the Redis database (using rladm upgrade db command) the Redis version will be updated to 3.0 even if you updated the cluster's Redis default version to 2.8.
Workaround: If you would like to cancel the old version indication in rladm status without upgrading the Redis version to 3.0 you should first change the cluster default version to 2.8 (using rladm tune cluster command), and then trigger the Redis process to be restarted by migrating the database shards (using rladm migrate db command).
- **Issue:** RLEC-8486 - On Ubuntu, when uninstalling RLEC using the apt-get purge command, some of the Redis processes on the machine might continue running.
Workaround: If you encounter this issue you must manually kill the Redis processes.
- **Issue:** RLEC-8283 - The cluster recovery process does not work properly when the cluster that needs to be recovered does not have a node with ID 1.
Workaround: If you encounter this issue please [contact Redis support](#)
- **Issue:** In the Replica Of process, if the target database does not have replication enabled and it is restarted or fails for any reason, the data on the target database might not be in sync with the source database, although the status of the Replica Of process indicates that it is.
Workaround: You must manually stop and restart the synchronization process in order to ensure the databases are in sync.
- **Issue:** In the Replica Of process, if the source database is resharded while the Replica Of process is active, the synchronization process will fail.
Workaround: You must manually stop and restart the synchronization process after the resharding of the source database is done.
- **Issue:** In the Replica Of process, high database traffic might restart the Replica Of process due to the "replica buffer" being exceeded. In this case you will often see the status of the Replica Of process display as "Syncing".
Workaround: You must manually increase the "replica buffer" size through rladm. In order to find the appropriate buffer size please [contact Redis support](#)
- **Issue:** In a cluster that is configured to support rack-zone awareness, if the user forces migration of a master or replica shard through rladm to a node on the same rack-zone as its corresponding master or replica shard, and later runs the rebalance process, the rebalance process will not migrate the shards to ensure rack-zone awareness compliance.
Workaround: In the scenario described above, you must use rladm to manually migrate the shard, to a node on a valid rack-zone in order to ensure rack-zone awareness compliance.

Updated: August 17, 2023

RLEC 4.2.1-30 Release Notes (October 18, 2015)

If you are upgrading from a previous version, make sure to review the [upgrade instructions](#) before running through the upgrade process.

New features

- rsyslog logging support - RLEC now writes by default to syslog, which enables monitoring through rsyslog.
- Profile support for tuning cloud and non-cloud environments – enables the administrator to configure RLEC to run with different performance profiles that are optimized for either cloud or non-cloud environments. For additional details, refer to the [Performance optimization](#) section.
- SLA for AOF rewrite - enables the administrator to configure database parameters, by running the radmin tune command, related to when AOF rewrite is triggered based on the time it would take to load the database from the AOF file, and on the maximum AOF rewrite file size. In addition, updates to the AOF rewrite mechanism minimize chances of the disk getting full.
- New warning and alerts related to AOF rewrite mechanism - a warning is shown during the setup process in case the disk size is lower than twice the size of the RAM. New cluster level alerts added to alert when node available disk space is lower than the needed disk space for AOF rewrite purposes, and when node performance is degraded due to reaching disk I/O limits.
- Replica Of support for multiple sources - the Replica Of feature is enhanced to support creating a database that is a replica of multiple source databases. For additional details, refer to the [Replica Of](#) section.
- Cross cluster Replica Of - the Replica Of feature now supports defining a database that is a replica of databases that belong to a different RLEC cluster. For additional details, refer to the [Replica Of](#) section.
- Multi-IP support - on a node that has multiple IPs, enables the administrator to specify which IP address is used for internal traffic and which IP addresses are used for external traffic. For additional details, refer to [Multi-IP & IPv6 support](#).
- IPv6 support for external traffic - on a node that has multiple IPs, external IP addresses can be of IPv6 type. For additional details, refer to [Multi-IP & IPv6 support](#) section.
- Support for OpenStack Object Store (“Swift”) location for import / export / backup. For additional details, refer to [Database backup](#) and [Importing data to a database](#) sections.
- Import of a sharded database - support for importing data of a sharded database by indicating multiple files paths. For additional details, refer to the [Importing data to a database](#) section.
- Enable running the install script in silent mode using “-y” parameter for default answers (“Y”) or “-c” for file path parameters for custom answers. For additional details, refer to [Accessing and installing the setup package](#) section.
- New radmin command-line-interface “info” command allows for fetching current value of tunable parameters.

Changes

- radmin command-line-interface can only be run under user root or redislabs. For additional details, refer to the [radmin command-line interface \(CLI\)](#) section.
- Import / export / backup to/from Amazon S3 requires supplying the credentials per usage instance; it does not use central cloud credentials that used to be supplied in the Settings -> General page.
- Fields related to storing Amazon S3 Cloud credentials have been removed from Settings -> General page.
- Performance optimization in the database resharding mechanism.
- Persistent and ephemeral storage cluster level alerts are enabled by default and set to 70%.
- Various enhancements to radmin command-line-interface (CLI) to support new commands.
- Redis version updated to 2.8.21 that addresses [CVE-2015-4335/DSA-3279 - Redis Lua Sandbox Escape](#).
- Port 3336 added to the list of ports being used by RLEC.
- Node “Network utilization” alert measured in percentages (%) has been updated to “Network throughput” alert measured in MBps. If the alert was defined then it will be disabled when upgrading to this version and the user needs to reconfigure it with a new value.
- Performance improvements to the database Import to make the process much faster.
- Enhancements to the support package to include additional details.
- Removed support for SSL 2.0/3.0 protocols due to security vulnerabilities.
- Disabled “dofile” functionality in Redis to solve a possible security vulnerability.
- radmin CLI “tune watchdog profile” command syntax updated to “tune cluster watchdog_profile”.

Fixed issues

- Support for relative path for backup and import functionalities.
- Fix issue with TLS configuration of email server settings.
- Email alerts not sent for database export and import processes.
- Fix erroneous entries in the Log page.
- Sometimes a wrong value is reported in the UI for node used ephemeral storage space.
- Sometime the wrong Replica Of Lag value is reported in the UI.
- RLEC-6875 - email server settings not working when using port 587.
- RLEC-5498 - Improve radmin response time when a node is down.
- Validation of Replica Of source definition did not fail in the UI and would only fail in runtime if it was using the Cluster Name (FQDN) and the FQDN was not properly configured in the DNS.
- Various improvements to error messages reported by radmin.

Known issues

- **Issue:** Connecting from a client to a database endpoint with mixed upper case and lower case letters can result in a slow response from the database.
Workaround: The cluster name (FQDN) should consist of only lower-case letters. When connecting from a client to a database endpoint, only lower case letters should be used in the endpoint.
- **Issue:** When upgrading a node to a new RLEC version (refer to [Upgrading nodes](#)) while the node is in the offline state (refer to [Taking a node offline](#), the

upgrade process succeeds but might result in an unstable cluster.

Workaround: Do not try to upgrade a node while it is in the offline state.

- **Issue:** In Red Hat Enterprise Linux, and CentOS operating systems, the process used for cleaning up internal log files does not work, thereby allowing the log files to grow and possibly result in disk space issues.

Workaround: On each machine that functions as a node in the cluster, create a file named “redislabs”, and save it in the following location: “/etc/logrotate.d/” (e.g. “/etc/logrotate.d/redislabs”). The file should contain the following text:

```
/var/opt/redislabs/log/*.log {
    daily
    missingok
    copytruncate
    rotate 7
    compress
    notifempty
}
```

The file's permissions should be root:root, 644. Afterwards, from the operating system command line interface (CLI) run the following commands:

```
yum install policycoreutils-python
semanage fcontext -a -t var_log_t '/var/opt/redislabs/log(/.*\*)?'
restsorecon -R /var/opt/redislabs/log
```

- **Issue:** In the Replica Of process, if the target database does not have replication enabled and it is restarted or fails for any reason, the data on target database might not be in sync with the source database, although the status of the Replica Of process indicates it is.

Workaround: You need to manually stop and restart the synchronization process in order to ensure the databases are in sync.

- **Issue:** In the Replica Of process, if the source database is resharded while the replica of process is active, the synchronization process will fail.

Workaround: You need to manually stop and restart the synchronization process after resharding of the source database is done.

- **Issue:** In the replica of process, high database traffic might restart the Replica of process as result of the “replica buffer” being exceeded. In this case you see the status of the replica of process as “Syncing” frequently.

Workaround: You need to manually reconfigure the “replica buffer” through rladmin and set the buffer size to a new size. In order to find the appropriate buffer size please contact support at: support@redislabs.com.

- **Issue:** In a cluster that is configured to support rack-zone awareness, if the user forces migration of a master or replica shard, through rladmin, to a node on the same rack-zone as its corresponding master or replica shard, and later runs the rebalance process, the rebalance process will not migrate the shards to ensure rack-zone awareness compliance.

Workaround: In the scenario described above, you need to manually migrate the shard, through rladmin, to a node on a valid rack-zone in order to ensure rack-zone awareness compliance.

Updated: August 17, 2023

RLEC 4.0.0-49 Release Notes (June 18, 2015)

If you are upgrading from a previous version, make sure to review the [upgrade instructions](#) before running through the upgrade process.

In addition, when running the install.sh script to upgrade the software, you might be prompted to approve changes to a configuration file named ccs-redis.conf. It is crucial that you choose Yes when asked whether to update this file.

New features

- Support for Red Hat Enterprise Linux (RHEL) and CentOS 6.5 and 7 operating systems.
- Support for additional AWS AMIs for Ubuntu and Amazon Linux, on multiple AWS regions.
- Support for additional browsers and operating systems for the management UI.
- Replica Of feature which enables creating a Redis database that keeps synchronizing data from another Redis database.
- Rack-zone awareness feature which enables mapping nodes to racks/zones to ensure a more sophisticated high-availability mechanism.
- Database-related alerts and email alerts.
- Auto-configuration of synchronization of cluster server clocks with NTP as part of installation script.
- Database Export functionality.

- Email alerts on database Export and Import.

Changes

- Database Backup Now functionality replaced with Export functionality.
- Database performance improvements to increase throughput and reduce latency.
- Improvement to AOF rewrite mechanism to deal with extreme-write scenarios and limited disk space configurations.
- Enhancements to radmin CLI to support additional commands.

Fixed issues

- Cluster stability improvements related to removing nodes and taking nodes offline.
- radmin CLI bug fixes.

Known issues

- **Issue:** RLEC-6819 - Uninstall on Red Hat Enterprise Linux does not stop all services and if you try to install the software again on the same machine the new installation might use prior installation data.

Workaround: Before installing the software again restart the machine or verify that all services are down.

- **Issue:** In the replica of process, if the source database is resharded while the replica of process is active, the synchronization process will fail.

Workaround: You need to manually stop and restart the synchronization process after resharding of the source database is done.

- **Issue:** In the replica of process, high database traffic might cause the replica of process to restart frequently as result of the “replica buffer” being exceeded. In this case you see the status of the replica of process as “Syncing” frequently.

Workaround: You need to manually reconfigure the “replica buffer” through radmin and set the buffer size to a new size. In order to find the appropriate buffer size please contact support at: support@redislabs.com.

- **Issue:** In a cluster that is configured to support rack-zone awareness, if the user forces migration of a master or replica shard, through radmin, to a node on the same rack-zone as its corresponding master or replica shard, and later runs the rebalance process, the rebalance process will not migrate the shards to ensure rack-zone awareness compliance.

Workaround: In the scenario described above, you need to manually migrate the shard, through radmin, to a node on a valid rack-zone in order to ensure rack-zone awareness compliance.

- **Issue:** In case you deploy a cluster and use the DNS option for the cluster name (see details in [How to set the Cluster Name \(FQDN\)](#), do not configure the DNS entries for the cluster nodes, and try to configure a database that is a replica of another database within the cluster, then the UI allows you to configure the source database but the replica of process fails in runtime.

Workaround: The configuration indicated in this issue is not a valid cluster configuration. If you choose to use the DNS option for the cluster name then you must configure DNS entries for the nodes, otherwise the cluster does not operate correctly. You have to either update the DNS accordingly, or recreate the cluster and use the mDNS option for the cluster name as described in [How to set the Cluster Name \(FQDN\)](#).

- **Issue:** When taking a node offline or removing a node, if the node being taken offline or removed is currently serving as the web server for the web browser being used to view the management UI, then the management UI appears down while the node is down.

Workaround: If you are using the cluster name in order to connect to the management UI in the browser, and the cluster name is registered in your external DNS or you are using the mDNS option, then the DNS entries will be updated to point to another node in the cluster after a few seconds and the UI will open properly. If you are not using the cluster name but rather the node IP in order to connect to the management UI in the web browser, you have to use the IP of another node in the cluster to access the management UI.

Updated: August 17, 2023

RLEC 0.99.5-24 Release Notes (February 15, 2015)

If you are upgrading from a previous version, make sure to review the [upgrade instructions](#) before running through the upgrade process.

If you are upgrading from version 0.99.5-11:

1. You must restart the services after the upgrade by running the following command with user root (sudo su). From the operating system’s CLI, run the following command: cnm_ctl restart
2. After the upgrade, radmin status command will report that the databases are from an old version. It is recommended that you upgrade the databases as soon as possible, as described in the [upgrade instructions](#).

New features

None.

Changes

- Enhancements to memtier_benchmark tool that is included in the installation package. You can find more details in the [GitHub project](#).

Fixed issues

- Improvements and fixes related to node failover, remove node and take node offline functionality.

Known issues

- **Issue:** When taking a node offline or removing a node, if the node being taken offline or removed is currently serving as the web server for the web browser being used to view the management UI, the management UI appears down while the node is down. **Workaround:** If you are using the cluster name in order to connect to the management UI in the browser, and the cluster name is registered in your external DNS or you are using the mDNS option, then the DNS entries will be updated to point to another node in the cluster after a few seconds and the UI will open properly. If you are not using the cluster name but rather the node IP in order to connect to the management UI in the web browser, you have to use the IP of another node in the cluster to access the management UI.

Updated: August 17, 2023

RLEC 0.99.5-11 Release Notes (January 5, 2015)

New features

Initial release, everything is new!

Changes

Initial release, no changes!

Fixed issues

None.

Known issues

- **Issue:** When taking a node offline or removing a node, if the node being taken offline or removed is currently serving as the web server for the web browser being used to view the management UI, the management UI appears down while the node is down. **Workaround:** If you are using the cluster name in order to connect to the management UI in the browser, and the cluster name is registered in your external DNS or you are using the mDNS option, then the DNS entries will be updated to point to another node in the cluster after a few seconds and the UI will open properly. If you are not using the cluster name but rather the node IP in order to connect to the management UI in the web browser, you have to use the IP of another node in the cluster to access the management UI.

Updated: June 16, 2021