Upgrade an existing Redis Enterprise Software deployment

To upgrade Redis Enterprise Software, you:

1. Upgrade the software on all nodes of the cluster.
2. (Optional) Upgrade each database in the cluster.

You don’t have to upgrade the databases in your cluster, however, new features and important fixes might not be enabled until you do so.

Supported upgrade paths

The upgrade path depends on two requirements, which vary according to the desired cluster version:

<table>
<thead>
<tr>
<th>Target cluster version</th>
<th>Minimum cluster version</th>
<th>Minimum database compatibility version</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>6.0.20</td>
<td>5.6</td>
<td>5.0</td>
</tr>
<tr>
<td>6.0</td>
<td>5.4.0</td>
<td>5.0</td>
</tr>
<tr>
<td>5.6</td>
<td>5.0.2-30</td>
<td>4.0</td>
</tr>
</tbody>
</table>

To upgrade successfully, both of the following must be true:

- Each node on your cluster must be at the minimum cluster version
- The Redis database compatibility version for each database must meet the required minimum.

If you do not meet these minimums, you must first update the nodes and databases accordingly.

Redis upgrade policy

In version 6.2.4, Redis Enterprise Software introduced the Redis database compatibility upgrade policy (redis_upgrade_policy). This policy controls the default value for the Redis database compatibility when creating or updating databases.

As of v6.2.4, this policy defaults to major, which limits Redis database compatibility to the most recent major release (v6.0, as of this writing).

This value supports a more conservative approach to upgrades. You can change the value to support more frequent upgrades, however, you’ll need to upgrade more frequently to stay current.

If you change the policy to latest, you need to upgrade Redis Enterprise Software every time there’s a minor release. Further, you’ll need to leave the policy set to latest until the next major release of Redis Enterprise Software, which generally happens every 18-24 months.

Changes to the upgrade policy do not affect existing databases. The policy is used only when you create a new database, upgrade a database, or change its configuration.
For best results, we recommend changing the policy value only after upgrading to a major release of Redis Enterprise Software.

**Upgrade policy values**

The Redis version of a database indicates its open source Redis compatibility. When you create a database or edit its configuration, the compatibility version is updated to support the most recent version supported by your copy of Redis Enterprise Software.

The database compatibility upgrade policy controls this by limiting compatibility to either the most recent major \( x.0 \) release or the latest release \( x.y \) supported by your version of Redis Enterprise Software.

- When set to `major`, the policy limits Redis compatibility to 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 limits compatibility to the latest (most recent) version of open source Redis supported by your copy of Redis Enterprise Software. (This was the default behavior of earlier releases. As of v6.2.4, 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.

To demonstrate: The Redis Enterprise Software 6.2.4 package included compatibility with the most recent major Redis release \( v6.0 \) at the time) and the latest (most recent) update to Redis \( v6.2.3 \) at the time).

By default, compatibility with \( v6.0 \) was installed with the upgrade.

To change this to use the latest release available, use `rladmin` to set the upgrade policy and the default Redis version:

```
tune cluster redis_upgrade_policy latest
```
```
tune cluster default_redis_version 6.2
```

**Upgrade a cluster**

**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.

- Identify the cluster master node and upgrade that node first.

  Use the `rladmin status nodes` command to identify the master node.

**Cluster upgrade process**

Starting with the master node, follow these steps for every node in the cluster. (We recommend upgrading each node separately to ensure cluster availability.)

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

   For help, see Download the installation package

2. Untar the installation package. Note that neither the installation path nor the user can be changed during the upgrade.
3. Verify node operation with the following commands:

```plaintext
rlcheck
rladmin status extra all
```

4. Run the install command:

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

The installation script automatically recognizes the upgrade and responds accordingly.

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

5. Verify node operation by repeating the commands from Step 3.

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

When each node is upgraded, the cluster is fully upgraded.

### Upgrade a database

#### 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.
- Use the `rladmin status all` command to display configuration details. (An indicator appears in the command output when the database compatibility version is out-of-date.)
- Verify the cluster is fully upgraded and operational.

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 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.

#### Database upgrade process

To upgrade a database:

1. Verify that the `redis_upgrade_policy` is set according to your preferences.

2. *(Optional)* Back up the database to minimize data loss.
3. Use `rladmin` to upgrade the database:

```bash
rladmin upgrade db <database name | database ID>
```

This restarts the database. No data is lost.

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

To do so:

- Use the admin console to open the Compatibility tab for the database; the Redis version displays the Redis database compatibility version.

- Use `rladmin status databases extra all` to display a list of the databases in your cluster and their current Redis database compatibility version.

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

When you upgrade an Active-Active (CRDB) database, you can also upgrade:

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

  The CRDB protocol is backward-compatible, which means v5.4.2 CRDB instances can understand write-operations from instances using the earlier CRDB protocol. However, earlier CRDB instances (those using the older protocol) cannot understand write-operations from instances using the newer protocol version.

  Once 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.

  **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; 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** - 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 of the database instances.

  **Guidelines:**

  - **As of v6.0.20,** feature version 0 is deprecated; 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.

To upgrade a CRDB instance:
1. Upgrade Redis Enterprise Software on each node in the clusters where the CRDB instances are located.

2. To see the status of your CRDB instances, run: `rladmin status`

   The statuses of the CRDB instances on the node can indicate:
   - OLD REDIS VERSION
   - OLD CRDB PROTOCOL VERSION
   - OLD CRBD FEATURESET VERSION

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

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

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

   ```bash
   [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]: [Y]
   ```

   The CRDB 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 CRDB instances. Then, to update the feature set for each active-active database, run:

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

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

   ```bash
   crdb-cli crdb list
   ```

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

<table>
<thead>
<tr>
<th>CRDB-GUID</th>
<th>NAME</th>
<th>REPL-ID</th>
<th>CLUSTER-FDQN</th>
</tr>
</thead>
<tbody>
<tr>
<td>700140c5-478e-49d7-ad3c-64d517ddc486</td>
<td>aatest</td>
<td>1</td>
<td>aatest1.example.com</td>
</tr>
<tr>
<td>700140c5-478e-49d7-ad3c-64d517ddc486</td>
<td>aatest</td>
<td>2</td>
<td>aatest2.example.com</td>
</tr>
</tbody>
</table>

Updated: October 31, 2021