### Multiple Active Proxy Support

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.

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.

### Proxy policies

A database can have one of these proxy policies:

<table>
<thead>
<tr>
<th>Proxy Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>There is only a single proxy that is bound to the database. This is the default database configuration and preferable in most use cases.</td>
</tr>
<tr>
<td>All Master Shards</td>
<td>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.</td>
</tr>
<tr>
<td>All Nodes</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

### Database configuration

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

**Warning:** Any configuration update which causes existing proxies to be unbounded can cause existing client connections to get
You can run rladmin 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.

```
$ rladmin 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 rladmin. 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.

```
rladmin 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 rladmin. 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:

```
rladmin 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 rladmin commands.

**Updated:** November 19, 2020