Create Active-Active databases on Kubernetes

On Kubernetes, Redis Enterprise Active-Active databases provide read and write access to the same dataset from different Kubernetes clusters. For more general information about Active-Active, see the Redis Enterprise Software docs.

Creating an Active-Active database requires routing network access between two Redis Enterprise clusters residing in two different Kubernetes clusters. Without the proper access configured for each cluster, syncing between the databases instances will fail.

This process consists of:

1. Documenting values to be used in later steps. It's important these values are correct and consistent.
2. Editing the Redis Enterprise cluster (REC) spec file to include the ActiveActive section. This will be slightly different depending on the K8s distribution you are using.
3. Creating the database with the crdb-cli command. These values must match up with values in the REC resource spec.

Prerequisites

Before creating Active-Active databases, you'll need two or more working Kubernetes clusters that each have:

- Routing for external access with an ingress controller (OpenShift users can use routes)
- A working Redis Enterprise cluster (REC) with a unique name
- Enough memory resources available for the database (see hardware requirements)

Document required parameters

The most common mistake when setting up Active-Active databases is incorrect or inconsistent parameter values. The values listed in the resource file must match those used in the crdb-cli command.

- **Database name** `<db-name>`:
  - Description: Combined with ingress suffix to create the Active-Active database hostname
  - Format: string
  - Example value: myaadb
  - How you get it: you choose
  - The database name requirements are:
    - Maximum of 63 characters
    - Only letter, number, or hyphen (-) characters
    - Starts with a letter; ends with a letter or digit.
    - Database name is not case-sensitive

You'll need the following information for each participating Redis Enterprise cluster (REC):

| Note: You'll need to create DNS aliases resolve your API hostname `<api-hostname>`, `<ingress-suffix>`, `<replication-hostname>` to the IP address for the ingress controller's LoadBalancer (or routes in OpenShift) for each database. To avoid entering multiple DNS records, you can use a wildcard in your alias (such as `*.ijk.redisdemo.com`). |

- **REC hostname** `<rec-hostname>`:
  - Description: Hostname used to identify your Redis Enterprise cluster in the crdb-cli command. This MUST be different from other participating clusters.
Format: `<rec-name>.<namespace>.svc.cluster.local`
Example value: `rec01.ns01.svc.cluster.local`

How to get it: List all your Redis Enterprise clusters

```
kubectl get rec
```

**API hostname** `<api-hostname>`:
- Description: Hostname used to access the Redis Enterprise cluster API from outside the K8s cluster
- Format: string
- Example value: `api.ijk.redisdemo.com`

**Ingress suffix** `<ingress-suffix>`:
- Description: Combined with database name to create the Active-Active database hostname
- Format: string
- Example value: `-cluster.ijk.redisdemo.com`

**REC admin credentials** `<username> <password>`:
- Description: Admin username and password for the REC stored in a secret
- Format: string
- Example value: `username: user@redisdemo.com, password: something`

How to get them:

```
kubectl get secret <rec-name> -o jsonpath='{.data.username}' | base64 --decode
kubectl get secret <rec-name> -o jsonpath='{.data.password}' | base64 --decode
```

**Replication hostname** `<replication-hostname>`:
- Description: Hostname used inside the ingress for the database
- Format: `<db-name><ingress-suffix>`
- Example value: `myaadb-cluster.ijk.redisdemo.com`
- How to get it: Combine `<db-name>` and `<ingress-suffix>` values you documented above.

**Replication endpoint** `<replication-endpoint>`:
- Description: Endpoint used externally to contact the database
- Format: `<db-name><ingress-suffix>:443`
- Example value: `myaadb-cluster.ijk.redisdemo.com:443`
- How to get it: `<replication-hostname>:443`

Add activeActive section to the REC resource file

From inside your K8s cluster, edit your Redis Enterprise cluster (REC) resource to add the following to the spec section. Do this for each participating cluster.

The operator uses the API hostname (`<api-hostname>`) to create an ingress to the Redis Enterprise cluster’s API; this only happens once per cluster. Every time a new Active-Active database instance is created on this cluster, the operator creates a new ingress route to the database with the ingress suffix (`<ingress-suffix>`). The hostname for each new database will be in the format `<db-name><ingress-suffix>`.

**Using ingress controller**

1. If your cluster uses an ingress controller, add the following to the spec section of your REC resource file.
2. After the changes are saved and applied, you can verify a new ingress was created for the API.

```bash
$ kubectl get ingress
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>HOSTS</th>
<th>ADDRESS</th>
<th>PORTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>rec01</td>
<td>api.abc.cde.redisdemo.com</td>
<td>225161f845b278-111450635.us.cloud.com</td>
<td>80</td>
<td>24h</td>
</tr>
</tbody>
</table>

3. Verify you can access the API from outside the K8s cluster.

```bash
curl -k -L -i -u <username>:<password> https://<api-hostname>/v1/cluster
```

If the API call fails, create a DNS alias that resolves your API hostname (<api-hostname>) to the IP address for the ingress controller’s LoadBalancer.

4. Make sure you have DNS aliases for each database that resolve your API hostname <api-hostname>, <ingress-suffix>, <replication-hostname> to the IP address of the ingress controller’s LoadBalancer. To avoid entering multiple DNS records, you can use a wildcard in your alias (such as *.ijk.redisdemo.com).

### Using Istio Gateway and VirtualService

No changes are required to the REC spec if you are using Istio in place of an ingress controller. The `activeActive` section added above creates ingress resources. The two custom resources used to configure Istio (Gateway and VirtualService) replace the need for ingress resources.

> Warning - These custom resources are not controlled by the operator and will need to be configured and maintained manually.

For each cluster, verify the VirtualService resource has two `- match:` blocks in the `tls` section. The hostname under `sniHosts:` should match your `<replication-hostname>`.

### Using OpenShift routes

1. Make sure your Redis Enterprise cluster (REC) has a different name (<rec-name .namespace>) than any other participating clusters. If not, you'll need to manually rename the REC or move it to a different namespace. You can check your new REC name with:

```bash
oc get rec -o jsonpath='{.items[0].metadata.name}'
```

If the rec name was modified, reapply `scc.yaml` to the namespace to reestablish security privileges.

```bash
oc apply -f scc.yaml
oc adm policy add-scc-to-group redis-enterprise-scc system:serviceaccounts:<namespace>
```

2. Make sure you have DNS aliases for each database that resolve your API hostname <api-hostname>, <ingress-suffix>, <replication-hostname> to the route IP address. To avoid entering multiple DNS records, you can use a
wildcard in your alias (such as *.ijk.redisdemo.com).

3. If your cluster uses OpenShift routes, add the following to the spec section of your Redis Enterprise cluster (REC) resource file.

   ```yaml
   activeActive:
     apiIngressUrl: <api-hostname>
     dbIngressSuffix: <ingress-suffix>
     method: openShiftRoute
   ```

4. Make sure you have a DNS aliases that resolve to the routes IP for both the API hostname (<api-hostname>) and the replication hostname (<replication-hostname>) for each database. To avoid entering each database individually, you can use a wildcard in your alias (such as *.ijk.redisdemo.com).

5. After the changes are saved and applied, you can see that a new route was created for the API.

   ```bash
   $ oc get route
   NAME    HOST/PORT                       PATH    SERVICES  PORT  TERMINATION
   WILDCARD rec01   api-openshift.apps.abc.redisdemo.com rec01   api             passthrough
   None
   ```

Create an Active-Active database with `crdb-cli`

The `crdb-cli` command can be run from any Redis Enterprise pod hosted on any participating K8s cluster. You’ll need the values for the required parameters for each Redis Enterprise cluster.

```bash
   crdb-cli crdb create
   --name <db-name> \
```

To create a database that syncs between more than two instances, add additional `--instance` arguments.

See the `crdb-cli reference` for more options.

**Test your database**

The easiest way to test your Active-Active database is to set a key-value pair in one database and retrieve it from the other.

You can connect to your databases with the instructions in Manage databases. Set a test key with `SET foo bar` in the first database. If your Active-Active deployment is working properly, when connected to your second database, `GET foo` should output `bar`.

Updated: March 28, 2022