Create replica databases on Kubernetes

You can configure a replica of a database by creating an item in the replicaSources section of the Redis Enterprise database specification. The value of replicaSourceType must be 'SECRET'; replicaSourceName must be the name of a secret that contains the replica source url.

A secret must be created using a stringData section containing the replica source URL as follows:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-replica-source
stringData:
  url: replica-source-url-goes-here
```

The replica source URL can be retrieved by going to "UI > database > configuration > Press the button Get Replica of source URL" in the administrative UI. But, this information can also be retrieved directly from the REST API as well.

A replica of database CR simply uses the secret in the replicaSources section:

```yaml
apiVersion: app.redislabs.com/v1alpha1
kind: RedisEnterpriseDatabase
metadata:
  name: name-of-replica
spec:
  redisEnterpriseCluster:
    name: name-of-cluster
  replicaSources:
    - replicaSourceType: SECRET
      replicaSourceName: my-replica-source
```

In the above, name-of-replica database will be created as a replica of the source database as long as the source database exists on the source cluster and the secret contains the correct replica source URL for that database.

Retrieving the replica source URL via kubectl

You will need kubectl, curl, and jq installed for this procedure.

1. Set your metadata:

   ```
   CLUSTER_NAME=test
   SOURCE_DB=db1
   TARGET_DB=db2
   TARGET_CLUSTER_NAME=test
   ```

2. Retrieve the cluster authentication:
3. Forward the port of the REST API service for your source cluster:

```
kubectl port-forward pod/${CLUSTER_NAME}-0 9443
```

4. Request the information from the REST API:

```
JQ='[[ | select(.name=="
JQ+="${SOURCE_DB}"
JQ+="") | ("redis://admin:" + .authentication_admin_pass + "@.endpoints[0].dns_name:"+(.endpoints[0].port|tostring))"
URI=`curl -sf -k -u "$CLUSTER_USER:$CLUSTER_PASSWORD" "https://localhost:9443/v1/bdbs?fields=uid,name,endpoints,authentication_admin_pass" | jq "$JQ" | sed 's/"//g'`
```

Note: URI now contains the replica source URI.

5. Construct the secret for the replica:

```
cat << EOF > secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: ${SOURCE_DB}-url
stringData:
  uri: ${URI}
EOF
kubectl apply -f secret.yaml
```

6. Create the replica database:

```
cat << EOF > target.yaml
apiVersion: app.redislabs.com/v1alpha1
kind: RedisEnterpriseDatabase
metadata:
  name: ${TARGET_DB}
spec:
  redisEnterpriseCluster:
    name: ${TARGET_CLUSTER_NAME}
  replicaSources:
  - replicaSourceType: SECRET
    replicaSourceName: ${SOURCE_DB}-url
EOF
kubectl apply -f target.yaml
```

### Automating the creation via a job

The following procedure uses a ConfigMap and a Job to construct the replica source URL secret from the source database and configure the target database.
There are four parameters:

- **source** - the name of the source database
- **cluster** - the name of the cluster for the source database
- **target** - the name of the target database
- **targetCluster** - the name of the cluster for the target database

These parameters can be set by:

```
kubectl create configmap replica-of-database-parameters \\   --from-literal=source=name-of-source \\   --from-literal=cluster=name-of-cluster \\   --from-literal=target=name-of-target \\   --from-literal=targetCluster=name-of-cluster
```

where "name-of-..." is replaced with the database source, source cluster, database target, and target cluster names.

The Job and ConfigMap below, when submitted, will create the secret and replica database:

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: replica-of-database
spec:
  backoffLimit: 4
  template:
    spec:
      serviceAccountName: redis-enterprise-operator
      restartPolicy: Never
      volumes:
      - name: scripts
        configMap:
          name: replica-of-database
      containers:
      - name: createdb
        image: debian:stable-slim
        env:
        - name: MY_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: SCRIPT
          value: create.sh
        - name: SOURCE_DB
          valueFrom:
            configMapKeyRef:
              name: replica-of-database-parameters
              key: source
        - name: TARGET_DB
          valueFrom:
            configMapKeyRef:
              name: replica-of-database-parameters
              key: target
```
- name: CLUSTER_SERVICE
t  value: .svc.cluster.local
- name: CLUSTER_NAME
t  valueFrom:
    configMapKeyRef:
      name: replica-of-database-parameters
      key: cluster
- name: CLUSTER_PORT
t  value: "9443"
- name: TARGET_CLUSTER_NAME
t  valueFrom:
    configMapKeyRef:
      name: replica-of-database-parameters
      key: targetCluster

volumeMounts:
- mountPath: /opt/scripts/
  name: scripts
  command:
  - /bin/bash
  - c
  - |
  apt-get update; apt-get install -y curl jq apt-transport-https gnupg2
  apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 6A030B21BA07F4FB
  curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add
  |
  echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | tee -a /etc/apt/sources.list.d/kubernetes.list
  apt-get update
  apt-get install -y kubectl
  bash /opt/scripts/$SCRIPT

---

apiVersion: v1
kind: ConfigMap
metadata:
  name: replica-of-database

data:
  create.sh:
    |
    CLUSTER_USER=`kubectl get secret/${CLUSTER_NAME} -o json | jq -r .data.username | base64 -d`
    CLUSTER_PASSWORD=`kubectl get secret/${CLUSTER_NAME} -o json | jq -r .data.password | base64 -d`
    CLUSTER_HOST=${CLUSTER_NAME}.${MY_NAMESPACE}${CLUSTER_SERVICE}
    JQ='.[[]] | select(.name=="'
    JQ+="${SOURCE_DB}"
    JQ+='") | ("redis://admin:" + .authentication_admin_pass + 
    @"+.endpoints[0].dns_name+"+(.endpoints[0].port|tostring))'
    URI=`curl -sf -k -u "$CLUSTER_USER:$CLUSTER_PASSWORD" https://${CLUSTER_HOST}:${CLUSTER_PORT}/v1/bdbs?fields=uid,name,endpoints,authentication_admin_pass" | jq "$JQ" | sed 's/"/\"g' 
    echo "URL: ${URL}"
    echo ""
    cat << EOF > /tmp/secret.yaml
    apiVersion: v1
    kind: Secret
    metadata:
      name: ${SOURCE_DB}-url
    stringData:
cat /tmp/secret.yaml
cat << EOF > /tmp/target.yaml
apiVersion: app.redislabs.com/v1alpha1
kind: RedisEnterpriseDatabase
metadata:
  name: ${TARGET_DB}
spec:
  redisEnterpriseCluster:
    name: ${TARGET_CLUSTER_NAME}
    replicaSources:
    - replicaSourceType: SECRET
      replicaSourceName: ${SOURCE_DB}-url
EOF
echo ---
cat /tmp/target.yaml
echo 
kubectl -n ${MY_NAMESPACE} apply -f /tmp/secret.yaml
kubectl -n ${MY_NAMESPACE} apply -f /tmp/target.yaml

Updated: October 6, 2021