REC custom resource options

A Redis Enterprise cluster (REC) is defined in a custom resource definition (CRD). The REC options are specified in the `spec` section of the custom resource. Any field not specified in the custom resource will be set to default values by the operator. Changes made to the custom resource will override changes made to the cluster via the admin console or `rladmin` commands.

The most common options you might specify are listed below. For a full list of options, see the Redis Enterprise Cluster API.

**name:** rec

This is the cluster name that the operator uses to name various resources in the Kubernetes cluster and also name the CRD.

**nodes:** nnn

This **must be an odd number** that is 3 or higher.

**uiServiceType:** service_type

This controls how the Redis Enterprise UI is exposed on the cluster. The `service_type` must be either `ClusterIP` or `LoadBalancer` (default: `ClusterIP`). It is an optional configuration based on k8s service types.

**persistentSpec**

You can add a `storageClassName` that specifies the Storage Class used for your nodes' persistent disks. For example, AWS uses "gp2" as a default, GKE uses "standard" and Azure uses "default".

Also, adding a `volumeSize` lets you control the size of the persistent volume attached to the Redis Enterprise pods.

```yaml
persistentSpec:
  enabled: true
  volumeSize: "10Gi"
  storageClassName: "gp2"
```

**redisEnterpriseNodeResources**

The compute resources required for each node (see limits and requests). Kubernetes accepts only integers as sizing numbers for requests and limits.

Resource limits are recommended to equal requests, see guaranteed quality of service for more info.

**limits**

The max resources (in integers) for a Redis node (similar to pod limits).

For example:
limits:
  cpu: "4000m"
  memory: 4Gi

The default is 4 cores (4000m) and 4GB (4Gi).

requests
The minimum resources (in integers) for a Redis node (similar to pod requests).
For example:

requests:
  cpu: "4000m"
  memory: 4Gi

The default is 4 cores (4000m) and 4GB (4Gi).

redisEnterpriseImageSpec
This configuration controls the Redis Enterprise version used, and where it is fetched from. This is an optional field. The Operator automatically uses the matching RHEL image version for the release.
The value is structured as follows with the policy values from OpenShift:

imagePullPolicy: IfNotPresent
Repository: redislabs/redis
versionTag: 6.0.6-39

The version tag is as it appears on your repository, such as in DockerHub.

redisUpgradePolicy
Redis upgrade policy that controls the default Redis database version when creating or updating databases.
The supported values are:

- major: limits Redis database version to the most recent major release
- latest: sets default database version to the latest minor release

More info:

- Redis upgrade policy
- Upgrade policy values

Sample REC custom resource
apiVersion: app.redislabs.com/v1
kind: RedisEnterpriseCluster
metadata:
  name: rec
  namespace: my-project
spec:
  nodes: 3
  persistentSpec:
    enabled: true
  uiServiceType: ClusterIP
  username: demo@redislabs.com
  redisEnterpriseNodeResources:
    limits:
      cpu: 4000m
      memory: 4Gi
    requests:
      cpu: 4000m
      memory: 4Gi
  redisEnterpriseImageSpec:
    imagePullPolicy: IfNotPresent
    repository: redislabs/redis
    versionTag: 6.0.6-39.rhel7-openshift

This example may be useful to get started with a test or development deployment. You can modify it for your required deployment use case. This file and other references are available in the RediLabs/redis-enterprise-k8s-docs GitHub repository.

Updated: May 11, 2022