Replica Set use Set-Based & Equity-Based selectors while replication controllers use Equity-Based selectors.

---

apiVersion: v1

kind: ReplicationController

metadata:

name: nginx

spec:

replicas: 3

selector:

app: nginx

template:

metadata:

name: nginx

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

---

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-rs-1

spec:

replicas: 3

selector:

# matchLabels:

# owner: sree

matchExpressions:

- { key: env, operator: In, values: [dev, prod] }

template:

metadata:

name: nginx

labels:

env: dev

spec:

containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80

---

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-rs-2

spec:

replicas: 3

selector:

# matchLabels:

# owner: sree

matchExpressions:

- { key: env, operator: In, values: [dev, prod] }

template:

metadata:

name: nginx

labels:

env: prod

spec:

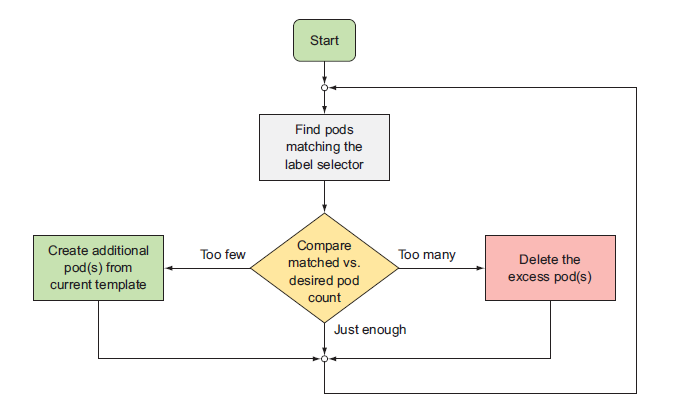
containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80



Creating a POD from KUBECTL

ku run --generator=run-pod/v1 testpod --image=index.docker.io/sreeharshav/rollingupdate:v3

1. If we change the label of the existing standalone pod similar to label of a RC, then the POD will fall under the RC scope.

2. The extra pods will be removed based on the timestamp and new PODs will be evicted/deleted.

https://medium.com/@zwhitchcox/matchlabels-labels-and-selectors-explained-in-detail-for-beginners-d421bdd05362

Only Job, Deployment, Replica Set, and Daemon Set support matchLabels.

kubectl rolling-update superstar superstar2 --image=index.docker.io/sreeharshav/testcontainer:v1

1. It will be a manual update from one image to other image.

2. New RC will be created and old RC will be deleted.

3. Roll back needs to change again to the old image.

4. Overall manual process and RC rolling-update is deprecated.

5. RC uses equality based labels like env: prod where as replica set uses set based labels as

{ key: env, operator: In, values: [dev, prod] }

kubectl rolling-update superstar2 superstar3 --image=index.docker.io/sreeharshav/rollingupdate:v1

kubectl rolling-update superstar2 superstar3 --rollback

kubectl rolling-update nginx nginx2 --image=index.docker.io/sreeharshav/rollingupdate:v3

kubectl rolling-update nginx2 nginx3 --image=index.docker.io/sreeharshav/rollingupdate:v5

root@ip-192-168-1-100:~# kubectl rolling-update nginx2 nginx3 --rollback

Command "rolling-update" is deprecated, use "rollout" instead

Error from server (NotFound): replicationcontrollers "nginx2" not found

### **Rolling updates**

The ReplicationController is designed to facilitate rolling updates to a service by replacing pods one-by-one.

As explained in [#1353](https://issue.k8s.io/1353), the recommended approach is to create a new ReplicationController with 1 replica, scale the new (+1) and old (-1) controllers one by one, and then delete the old controller after it reaches 0 replicas. This predictably updates the set of pods regardless of unexpected failures.

Ideally, the rolling update controller would take application readiness into account, and would ensure that a sufficient number of pods were productively serving at any given time.

The two ReplicationControllers would need to create pods with at least one differentiating label, such as the image tag of the primary container of the pod, since it is typically image updates that motivate rolling updates.