Lab - Labels,Selectors,ReplicationController and replicaset in Kubernetes

08:55 PM

- Labels are key-value pairs which are attached to pods, replication controller and services.
- They are used as identifying attributes for objects such as pods and replication controller.
- They can be added to an object at creation time and can be added or modified at the run time.
- With **Label Selectors** we can select a subset of objects. Kubernetes supports two types of Selectors:

Equality-Based Selectors

- Equality-Based Selectors allow filtering of objects based on label keys and values. With this type of Selectors, we can use the =, ==, or **!=**operators. For example, with **env==dev** we are selecting the objects where the **env** label is set to **dev**.

Equality Based Requirement

- Equality based requirement will match for the specified label and filter the resources. The supported operators are =, ==, !=

adpspl-mac36:~ vbirada$ kubectl get pod --show-labels

```
NAME          READY   STATUS    RESTARTS   AGE     LABELS
example-pod   1/1     Running   0          17h
env=prod,owner=Ashutosh,status=online,tier=backend

example-pod1  1/1     Running   0          21m
env=prod,owner=Shovan,status=offline,tier=frontend

example-pod2  1/1     Running   0          8m
env=dev,owner=Abhishek,status=online,tier=backend

example-pod3  1/1     Running   0          7m
env=dev,owner=Abhishek,status=online,tier=frontend
```

Now, I want to see all the pods with online status:

adpspl-mac36:~ vbirada$ kubectl get pods -l status=online

```
NAME          READY   STATUS    RESTARTS   AGE
example-pod   1/1     Running   0          17h
example-pod2  1/1     Running   0          9m
example-pod3  1/1     Running   0          9m
```

adpspl-mac36:~ vbirada$ kubectl get pods -l status!=online

```
NAME          READY   STATUS    RESTARTS   AGE
example-pod1  1/1     Running   0          25m
example-pod4  1/1     Running   0          11m
```

adpspl-mac36:~ vbirada$ kubectl get pods -l status==offline

```
NAME          READY   STATUS    RESTARTS   AGE
example-pod1  1/1     Running   0          26m
example-pod4  1/1     Running   0          11m
```

adpspl-mac36:~ vbirada$ kubectl get pods -l status==offline,status=online

No resources found.

adpspl-mac36:~ vbirada$ kubectl get pods -l status==offline,env=prod

```
NAME          READY   STATUS    RESTARTS   AGE
example-pod1  1/1     Running   0          28m
```

adpspl-mac36:~ vbirada$ kubectl get pods -l owner=Abhishek

```
NAME          READY   STATUS    RESTARTS   AGE
example-pod2  1/1     Running   0          15m
example-pod3  1/1     Running   0          14m
```

In above commands, labels separated by comma is a kind of **AND** satisfy operation.

Set-Based Selectors

- Set-Based Selectors allow filtering of objects based on a set of values. With this type of Selectors, we can use the **in**, **notin**, and **exist** operators. For example, with **env in (dev,qa)**, we are selecting

objects where the **env** label is set to **dev** or **qa**.

Set Based Requirement

- Label selectors also support set based requirements. In other words, label selectors can be used to specify a set of resources.
- The supported operators here are in , notin and exists .

adpspl-mac36:~ vbirada$ kubectl get pod -l 'env in (prod)'
NAME READY STATUS RESTARTS AGE
example-pod 1/1 Running 0 18h
example-pod1 1/1 Running 0 41m
adpspl-mac36:~ vbirada$ kubectl get pod -l 'env in (prod,dev)'
NAME READY STATUS RESTARTS AGE
example-pod 1/1 Running 0 18h
example-pod1 1/1 Running 0 41m
example-pod2 1/1 Running 0 27m
example-pod3 1/1 Running 0 27m
Here env in (prod,dev) the comma operator acts as a OR operator. That is it will list pods which are
in prod or dev.

Kubectl Get pods --show-labels

EXAMPLE OF LABELS
Try on label scenarios

```
kind: Pod
apiVersion: v1
metadata:
  name: delhipod
  labels:
    env: development
    class: pods
spec:
    containers:
      - name: c00
        image: ubuntu
        command: ["/bin/bash", "-c", "while true; do echo Hello-world; sleep 5 ; done"]
```

****
NODE SELECTOR EXAMPLE (we can use this on some cases)

Which node to select out of all worker node
We first have to apply label on node -

```
kind: Pod
apiVersion: v1
metadata:
  name: nodelabels
  labels:
    env: development
spec:
    containers:
      - name: c00
        image: ubuntu
        command: ["/bin/bash", "-c", "while true; do echo Hello-world; sleep 5 ; done"]
    nodeSelector:
      hardware: t2-medium
```

Kubectly apply -f pod.yml

Kubectl get pods

Descibe pod

Cubectl get nodes

Kubectl label nodes nameofnode hardware=t2-medium # to apply label to node

Delete it

****

Scaling
Replication

Can't start stopped pod-- but can create same config pods

Replicas=2 // similar config pod will get created

Senarios:
Rc = replicas: 4
Ran it started rc - it create 4 pods which is controlled by rc
 # labels ---> going to add on pods are very very important

If labels added on pod get marched with selector applied on RC object then RC Object going to manage or consider that pod to maintain **d.s of that controller**

We modified our first.yaml --- added same label which is selector for RC
Kubectl apply -f first.yaml

We delete 1 pod--> **A.S 3 !== D.S 4**
Try maintain it by creating new pod. Scanned pods

Instade of creating new pods from scratch it started considering that pod is part of R.C

EXAMPLE OF REPLICATION CONTROLLER
If pod failed, crashed, terminated it will keep desire state

```
kind: ReplicationController        # object kind RC
apiVersion: v1
metadata:
  name: myreplica #name of object
spec:
  replicas: 3      # how many pods need to create
  selector:        # which pod to watch/belong to rc
    myname: Surekha     # much match the label
  template:      # what will be inside pod that we defind on template
    metadata:
      name: testpod6 # name of pof
      labels:        # label on pof
        myname: Surekha
    spec: # what will be inside pod / contaier inside pof
      containers:
        - name: c00
          image: ubuntu
          command: ["/bin/bash", "-c", "while true; do echo Hello-wolld ; sleep 5 ; done"]
```

Kubectl scale --replicas=8 rc -l myname=Surekha // to scale pod
Kubectl scale --replicas=3 rc -l myname=Surekha // to scale down pod

Kubectl delete -f myrc.yml

****

The only difference between replica set and replication controller is the selector types. The replication controller supports equality based selectors whereas the replica set supports equality based as well as set based selectors

EXAMPLE OF REPLICA SET

Replica set is a next generation of replication controller
Replication controller supports equality based selector where as
RS supports set based selection - (filtering according to set of values)

ReplicaSet

- The ReplicaSet controller simply ensures that the desired number of pods matches its **label selector** exists and are operational
- If the labels of the pod are modified and they do not match the label selector, then a new pod is spawned, the old one stays there.
- The ReplicaSet provide a declarative definition of what a Pod should be and how many of it should be running at a time.

rs1.yaml

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

Working with ReplicaSet

Create the ReplicaSet
- kubectl create -f rs1.yaml

Check the status
- kubectl get rs [--watch]
- kubectl describe rs nginx

Change the number of replicas
- kubectl scale rs nginx --replicas=3

Delete the ReplicaSet
- kubectl delete rs nginx

```
kind: ReplicaSet
apiVersion: apps/v1
metadata:
  name: myrs
spec:
  replicas: 2
  selector:
    matchExpressions:                    # these must match the labels
      - {key: myname, operator: In, values: [surekha, shelake, foo]}
      - {key: env, operator: NotIn, values: [production]}
  template:
    metadata:
      name: testpod7
      labels:
        myname: surekha
    spec:
     containers:
       - name: c00
         image: ubuntu
         command: ["/bin/bash", "-c", "while true; do echo Hello world; sleep 5 ; done"]
```