55. Labels and Selectors - practice & learn clearly

Labels:- To identify and group

what
> It is a key-value pairs ("key = value") attached to k8s objects (Pods, Deployments etc.,)

why
> Identify and group related resources (eg: env=prod, app=nginx)

Eg:- It will be placed under metadata

```
metadata:
  labels:
    app: nginx
    env: production
```

Selectors To find and manage

what
> Selectors used to find and manage resources based on their labels

why
> Enable connections b/w objects (e.g. a Service selects to route traffic)

Eg:-
```
selector:
  matchLabels:
    app: nginx
```
* Targets all pods with "app=nginx"

Labels → Like sticky notes on a book ("Sci-Fi", "Best-seller")

Selectors → Like a librarian finding all books with "Sci-Fi" note

Labels identify
Selectors find

How labels & selectors works in K8S

> We've created alot of objects in K8S like pods, Services, Replica sets, Deployments, etc.,

> For K8S, all of these are different objects, overtime we've 100's & 1000's of these objects in your cluster then you will need a way to filter & view different objects by different categories, such as

* to group objects by their type (or)
* View objects by application (or)
* By their functionality.

whatever it maybe, you can group and Select objects using labels and selectors

For each app, attach labels as per their needs like apps, functions etc.,

app: app1
function: Front-end

app: app2
function: Back-end

app: app3
function: web-servers.

app: app4
function: DB

app: app5
function: Image-proc

While selecting, specify a condition to filter specific objects

> In pod-definition file under "metadata" add labels in key: value pair format

> We can add as many as labels you want

pod.definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
    name: simple-webapp
    labels:
        app: App1
        function: Front-end

spec:
    containers:
    - name: simple-webapp
      image: simple-webapp
      ports:
        - containerPort: 8080
```

Once the pod is created, to select the pod with the labels use "selector" option.

```
kubectl get pods --selector app=App1
```

NAME	READY	STATUS	RESTARTS	AGE
simple-webapp	0/1	Completed	0	1d

> K8S objects use labels and selectors internally to connect different object together

For eg:-

To create a "replicaset" consisting of 3 different pods, we first label the "pod.definition" & use Selector in a "replicaset" to group the pods

In "replicaset-definition.yaml" we can see labels in 2 places

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: simple-webapp
  labels:                          # labels for the replicaset
    app: App1
    function: Front-end
spec:
  replicas: 3
  selector:
    matchLabels:
      app: App1
  template:
    metadata:
      labels:                      # labels configured for the pod
        app: App1
        Function: Front-end
    spec:
      containers:
        - name: Simple-webapp
          image: Simple-webapp
```

definition file
metadata

Explanation for the leftside Replicaset definition file.

> The labels on the replicaset will be used, you where to configure some other object to discover the replicaset [For eg- services]

> Anyway to connect, replicaset to the pod, we config the selector field under the replica set specificati to match the labels defined on the pod

> A single label will do if it matches correctly, However if you feel there could be other pods with the same label but with a different function. the we could specify both the labels to ensure that the right pods are discovered by the replicaset

> On creation if labels match, the replicaset created successfully, it works the same for other obj like a service

When a service is created, it uses the selector defined in the "service definition file" to match the labels set on the pods in the "replicaset definition f

service-definition.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: App1
  ports:
  - protocols: TCP
    port: 80
    targetPort: 9376
```

Finally, lets look at annotations, while "labels" a

"Selectors" are used to group and "select" objects, "annotations" are used to record other details for informatory purpose.

for eg:- Tool details, like name, version, build info. etc., or contact details, phone numbers, email IDs, etc., that may be used for some kind of integration purpose

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: simple-webapp  # name for ReplicaSet
  labels:              # label for the Replicaset - simple-webapp
    app: App1
    function: Front-end
  annotations:
    buildVersion: 1.34
spec:
  replicas: 3
  selector:
    matchLabels:
      app: App1
  template:
    metadata:
      labels:
        app: App1
        function: Front-end
    spec:
      containers:
      - name: simple-webapp
        image: simple-webapp
```

We've labelling the object of the "kind: Replicaset" whose name is "simple-webapp"

The Replicaset has to select the pod which has this label "app: App1"

Both should match

pod definition file

Heal ready objects

Debug:- The label under spec, selector, matchlabel for the replicaset should match with the label under template

Fixed ReplicaSet Definition.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend-replicaset

spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
      tier: web
  template:
    metadata:
      labels:
        app: frontend
        tier: web

    spec:
      containers:
      - name: nginx-container
        image: nginx:1.17
```

Both should match.

(7) what would happen if you've changed the image version in the pod template after creation?

Ans:- The change wouldn't affect existing pods. you would need to delete existing pods for the Replicaset to recreate them with the new image

Instances where labels can be changed & whose not

Instances where labels can be changed.

* Standalone Pods.

You can freely add, update (or remove) labels (on a running pod using kubectl label. This change is immediate & persists after the pod's lifetime.

* Pods managed by Controllers [Deployments, Replicasets etc.,]

You can change labels on individual pods, but those changes are temporary. If the pod is deleted/recreated by the Controller (e.g. during scaling or during rolling updates) then new pod will have labels defined in the Controller's pod template, not your manual changes.

Best practice: Update the label in the Controllers pod template for persistent changes.

* Using --overwrite flag:

If a label already exists, you must use the --overwrite flag to change its value.

Instances where labels cannot (or should not) be changed

* Immutable or restricted labels

Some labels may be restricted by cluster policies or admission controllers, preventing changes for compliance or security reason. attempting to change these will result in an error.

* System/Reserved labels:

Labels with reserved prefixes (like "kubernetes.io/" or "kgs.ro/") are intended for system use. Modifying these is not recommended and may be blocked or causes issues.

* pods that are terminating or terminated

You cannot change labels on pods that are terminating or have already terminated

* Pod template labels in Controllers

You cannot change the labels of a Controller's pod template (the template that defines how new pods are created) on a running pod. You must update the Controllers YAML & reapply it direct changes to the template on a running pod are not allowed.