Labels and Selectors

Introduction:

Labels play important part in Kubernetes. Labels are in the form of key and value that are attached to objects, such as pods. Deployment etc. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users.

Objectives:

1. Assigning labels to the pods or deployment
2. Selecting on the basis of labels

1. Assigning labels to the pods or deployment:

we are creating a deployment with multiple labels which is shown below.

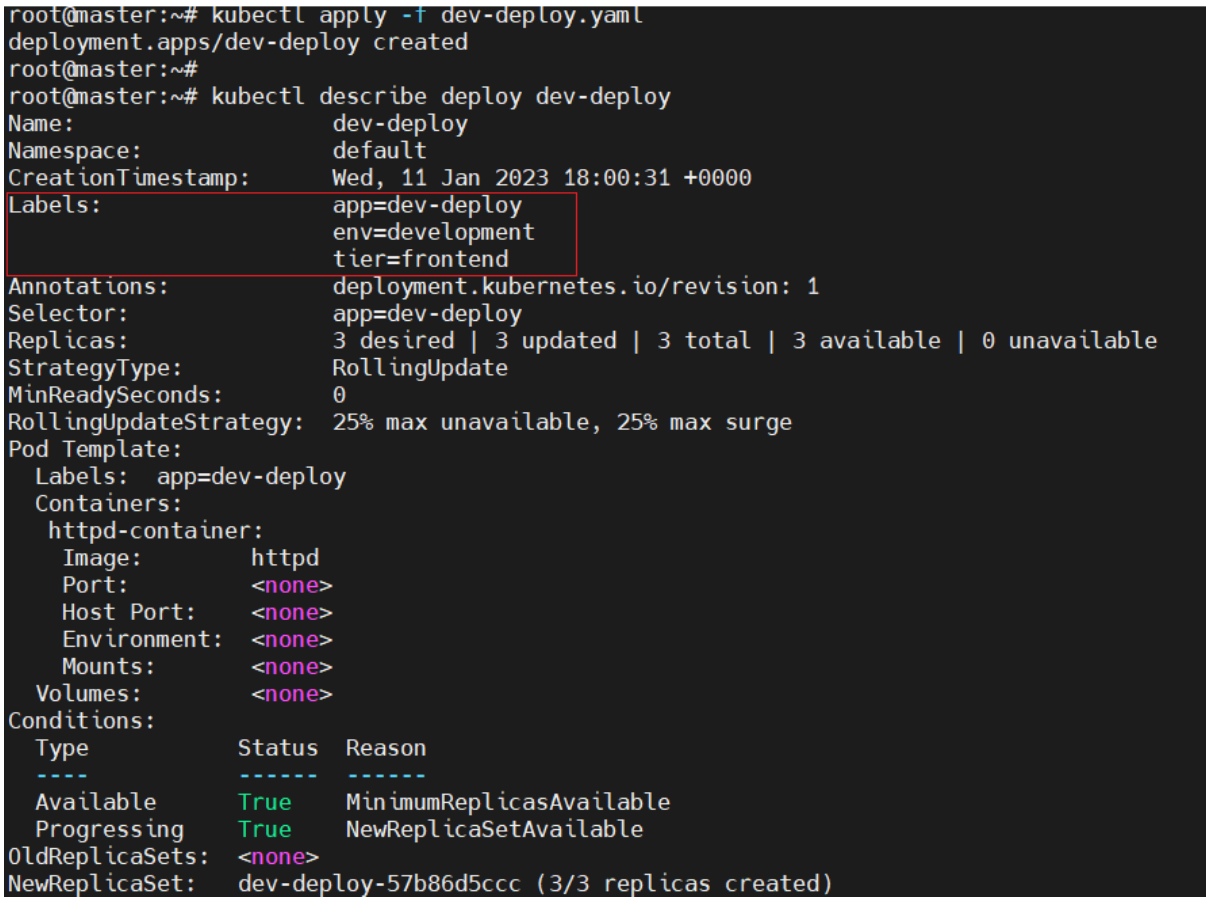
```
# vi dev-deploy.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dev-deploy
    env: development
    tier: frontend
  name: dev-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: dev-deploy
  template:
    metadata:
      labels:
        app: dev-deploy
    spec:
      containers:
      - image: httpd
        name: httpd-container
```

Now apply the definition file using the below command.

```
Kubectl apply -f dev-deploy.yaml
```

We can see that these labels have been attached to deployment as well as the pods related to this deployment.



```
root@master:~# kubectl apply -f dev-deploy.yaml
deployment.apps/dev-deploy created
root@master:~#
root@master:~# kubectl describe deploy dev-deploy
Name:                   dev-deploy
Namespace:              default
CreationTimestamp:      Wed, 11 Jan 2023 18:00:31 +0000
Labels:                 app=dev-deploy
                        env=development
                        tier=frontend
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=dev-deploy
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=dev-deploy
  Containers:
   httpd-container:
    Image:        httpd
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   dev-deploy-57b86d5ccc (3/3 replicas created)
```

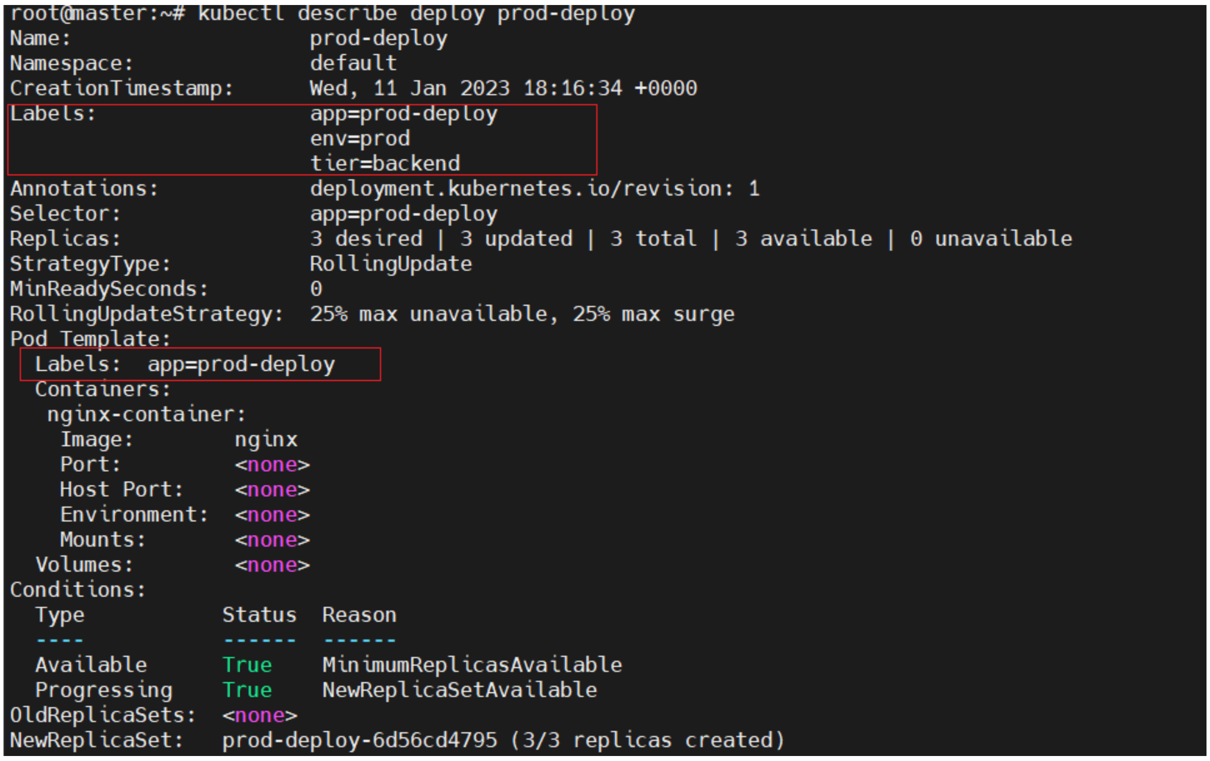
Let’s create another deployment with different labels.

```
# vi prod-deploy.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: prod-deploy
    env: prod
    tier: backend
  name: prod-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: prod-deploy
  template:
    metadata:
      labels:
        app: prod-deploy
    spec:
      containers:
      - image: nginx
        name: nginx-container
```

Now apply the changes using the below command and see the result.

```
Kubectl apply -f prod-deploy.yaml
```



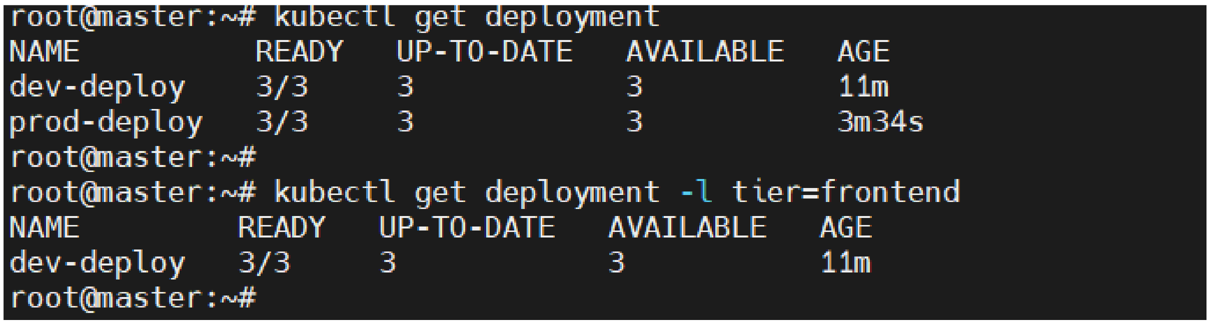
```
root@master:~# kubectl describe deploy prod-deploy
Name:                   prod-deploy
Namespace:              default
CreationTimestamp:      Wed, 11 Jan 2023 18:16:34 +0000
Labels:                 app=prod-deploy
                        env=prod
                        tier=backend
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=prod-deploy
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=prod-deploy
  Containers:
   nginx-container:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   prod-deploy-6d56cd4795 (3/3 replicas created)
```

2. Selecting on the basis of labels:

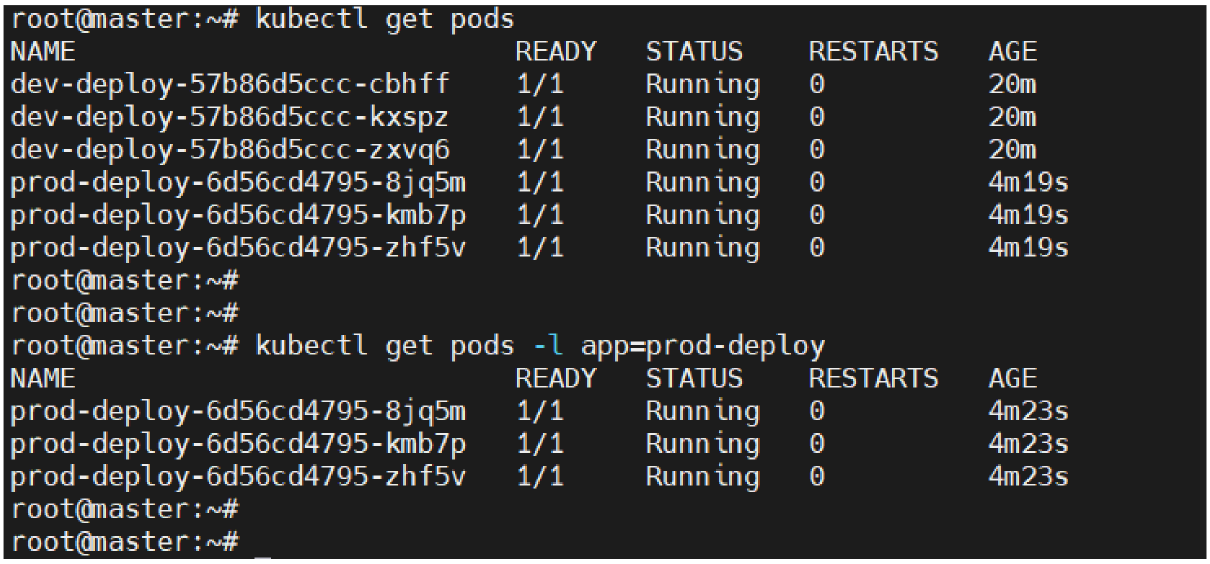
Now we have 2 deployments running in our cluster having different labels attached to it.

We can get the pods or deployments using the labels filter option. Use the below commands to filter them.

```
kubectl get deployment -l tier=frontend
kubectl get pods -l app=prod-deploy
```



```
root@master:~# kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
dev-deploy    3/3     3            3           11m
prod-deploy   3/3     3            3           3m34s
root@master:~#
root@master:~# kubectl get deployment -l tier=frontend
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
dev-deploy   3/3     3            3           11m
root@master:~#
```



```
root@master:~# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
dev-deploy-57b86d5ccc-cbhff    1/1     Running   0          20m
dev-deploy-57b86d5ccc-kxspz    1/1     Running   0          20m
dev-deploy-57b86d5ccc-zxvq6    1/1     Running   0          20m
prod-deploy-6d56cd4795-8jq5m   1/1     Running   0          4m19s
prod-deploy-6d56cd4795-kmb7p   1/1     Running   0          4m19s
prod-deploy-6d56cd4795-zhf5v   1/1     Running   0          4m19s
root@master:~#
root@master:~#
root@master:~# kubectl get pods -l app=prod-deploy
NAME                           READY   STATUS    RESTARTS   AGE
prod-deploy-6d56cd4795-8jq5m   1/1     Running   0          4m23s
prod-deploy-6d56cd4795-kmb7p   1/1     Running   0          4m23s
prod-deploy-6d56cd4795-zhf5v   1/1     Running   0          4m23s
root@master:~#
root@master:~# _
```