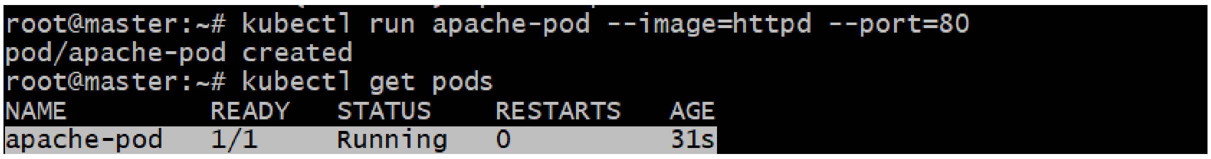
K8s 03

1) Create a ClusterIP service for an Apache web server pod.

→First create a pod with name apache-pod using below command:

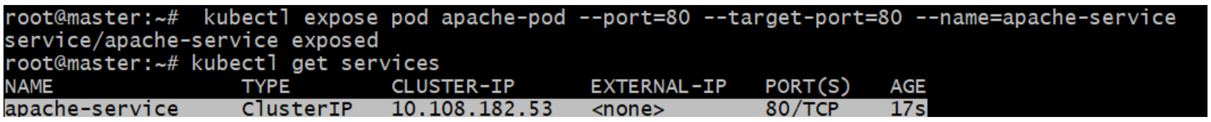
kubectl run apache-pod --image=httpd --port=80



```
root@master:~# kubectl run apache-pod --image=httpd --port=80
pod/apache-pod created
root@master:~# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
apache-pod    1/1     Running   0          31s
```

→now create a clusterIP service for apache-pod by using below command:

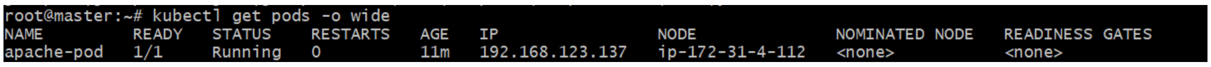
kubectl expose pod apache-pod --port=80 --target-port=80 --name=apache-service (here type taken as default no need mention type in command)



```
root@master:~#  kubectl expose pod apache-pod --port=80 --target-port=80 --name=apache-service
service/apache-service exposed
root@master:~# kubectl get services
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
apache-service   ClusterIP   10.108.182.53   <none>        80/TCP    17s
```

→publically not exposed its work only internally:

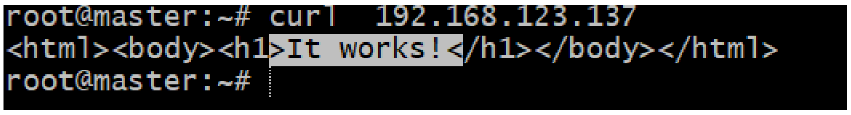
-- First do **kubectl get pods -o wide**



```
root@master:~# kubectl get pods -o wide
NAME         READY   STATUS    RESTARTS   AGE   IP                NODE              NOMINATED NODE   READINESS GATES
apache-pod   1/1     Running   0          11m   192.168.123.137   ip-172-31-4-112   <none>           <none>
```

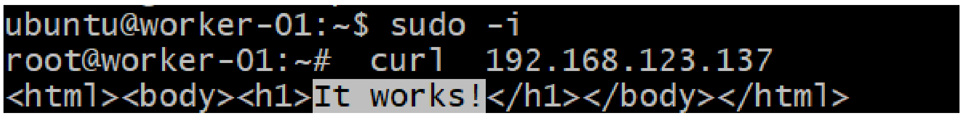
--then check it internally in master

curl 192.168.123.137



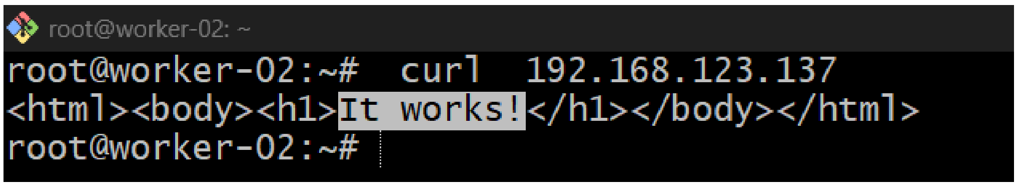
```
root@master:~# curl  192.168.123.137
<html><body><h1>It works!</h1></body></html>
root@master:~#
```

In worker-01



```
ubuntu@worker-01:~$ sudo -i
root@worker-01:~#  curl  192.168.123.137
<html><body><h1>It works!</h1></body></html>
```

In worker-02



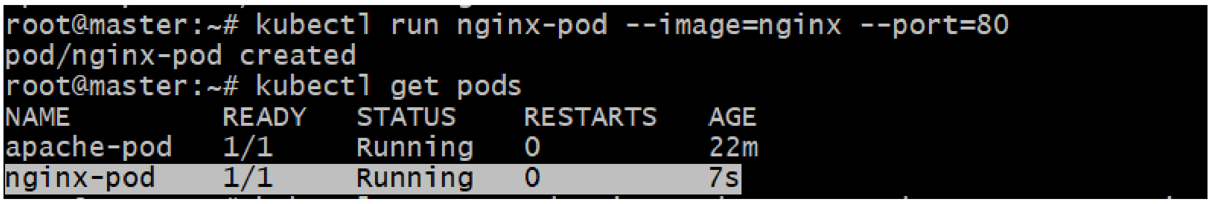
```
root@worker-02: ~
root@worker-02:~#  curl  192.168.123.137
<html><body><h1>It works!</h1></body></html>
root@worker-02:~#
```

2) Expose an Nginx pod externally using a NodePort service.

→First create nginx pod by using below command:

kubectl run nginx-pod --image=nginx --port=80

check it

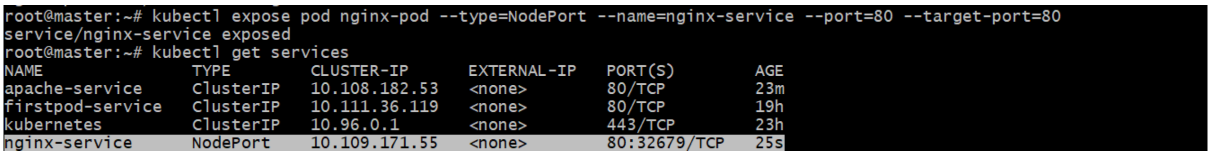


```
root@master:~# kubectl run nginx-pod --image=nginx --port=80
pod/nginx-pod created
root@master:~# kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
apache-pod   1/1     Running   0          22m
nginx-pod    1/1     Running   0          7s
```

→now Expose an Nginx pod externally using a NodePort service by using below command:

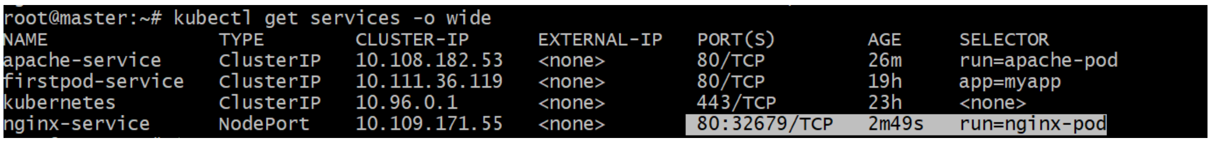
kubectl expose pod nginx-pod --type=NodePort --name=nginx-service --port=80 --target-port=80

check it…Node port service has been created



```
root@master:~# kubectl expose pod nginx-pod --type=NodePort --name=nginx-service --port=80 --target-port=80
service/nginx-service exposed
root@master:~# kubectl get services
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
apache-service     ClusterIP   10.108.182.53   <none>        80/TCP         23m
firstpod-service   ClusterIP   10.111.36.119   <none>        80/TCP         19h
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP        23h
nginx-service      NodePort    10.109.171.55   <none>        80:32679/TCP   25s
```

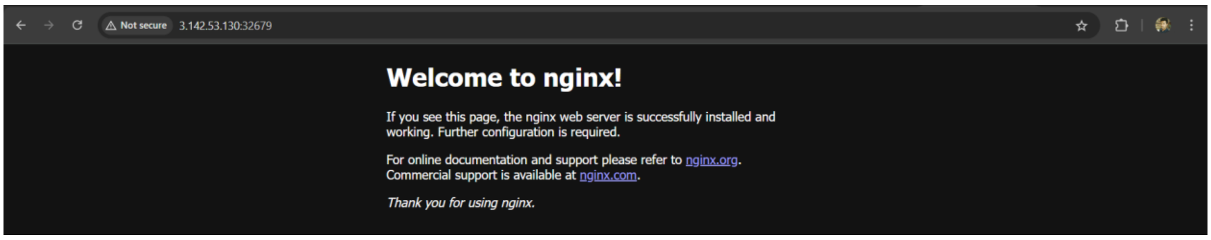
kubectl get services -o wide



```
root@master:~# kubectl get services -o wide
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE    SELECTOR
apache-service     ClusterIP   10.108.182.53   <none>        80/TCP         26m    run=apache-pod
firstpod-service   ClusterIP   10.111.36.119   <none>        80/TCP         19h    app=myapp
kubernetes         ClusterIP   10.96.0.1       <none>        443/TCP        23h    <none>
nginx-service      NodePort    10.109.171.55   <none>        80:32679/TCP   2m49s  run=nginx-pod
```

→now check in browser:

Checked with master pubip:32679



Not secure 3.142.53.130:32679

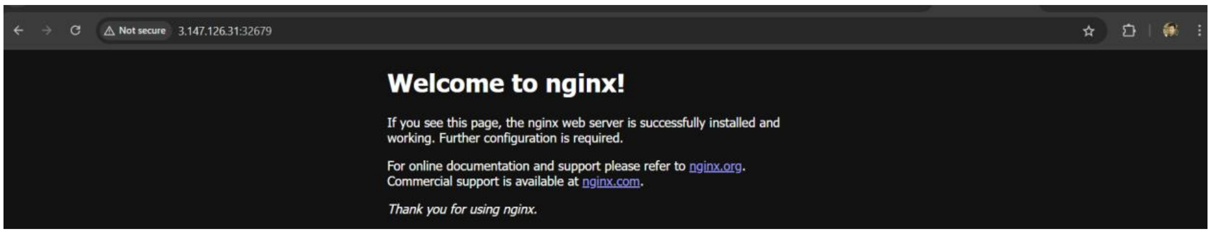
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Checked with worker-01 pubip:32679



Not secure 3.147.126.31:32679

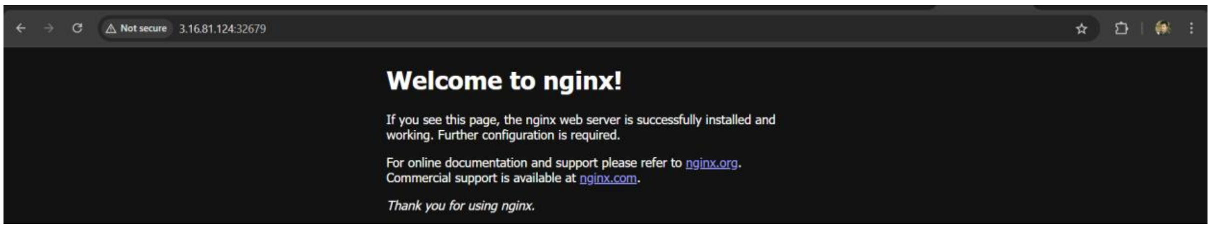
Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Checked with worker-02pubip:32679



Not secure 3.16.81.124:32679

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

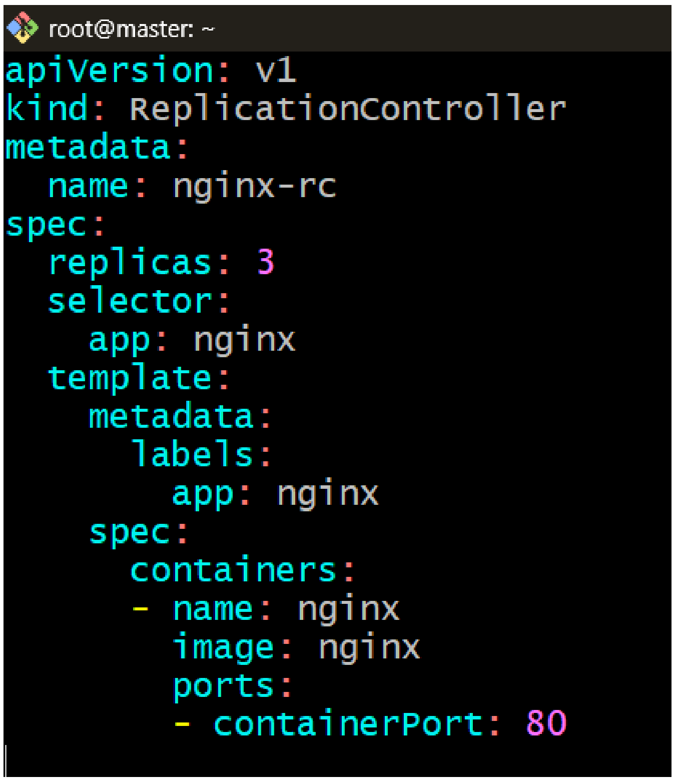
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

3) Deploy a ReplicationController to maintain 3 replicas of an Nginx pod.

→ Create ReplicationController using YAML file:

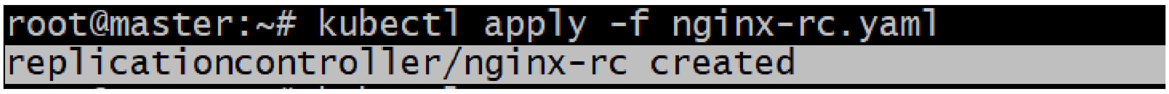
```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```



```
root@master: ~
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

→run the yaml file using below command to create rc:

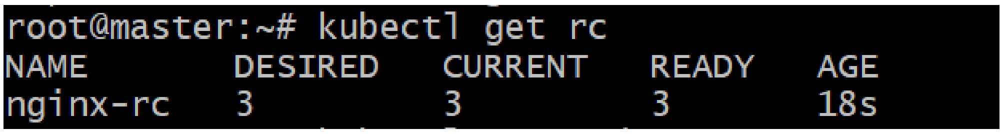
kubectl apply -f nginx-rc.yaml



```
root@master:~# kubectl apply -f nginx-rc.yaml
replicationcontroller/nginx-rc created
```

--Rc created

kubectl get rc

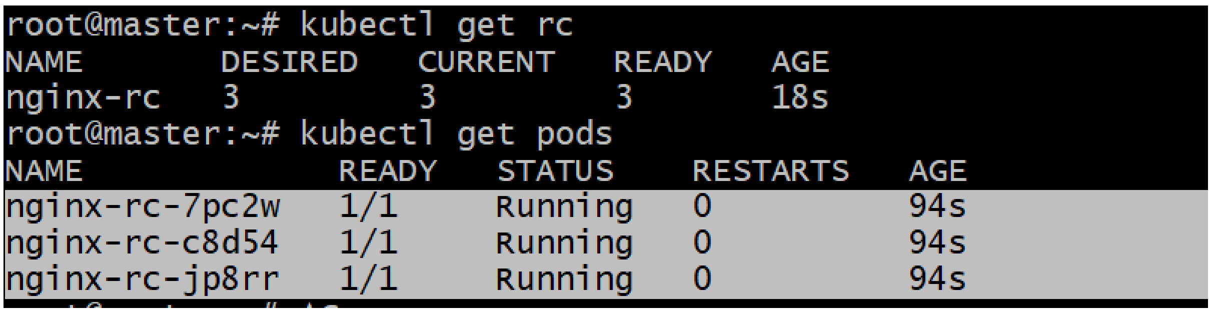


```
root@master:~# kubectl get rc
NAME       DESIRED   CURRENT   READY   AGE
nginx-rc   3         3         3       18s
```

→now ReplicationController created to maintain 3 replicas of an Nginx pod:

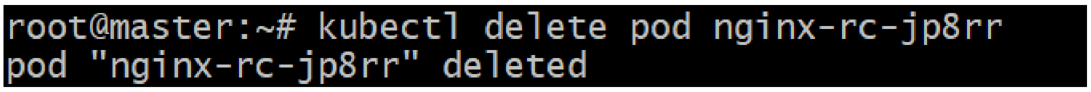
Check with this command

kubectl get pods

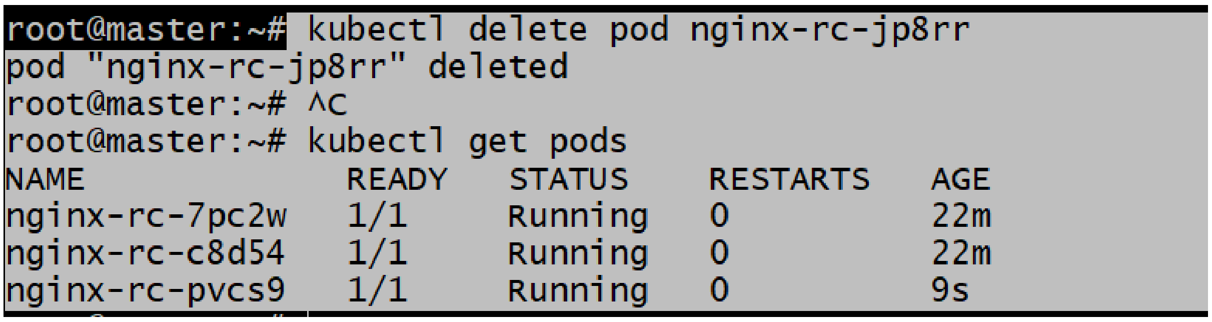


```
root@master:~# kubectl get rc
NAME       DESIRED   CURRENT   READY   AGE
nginx-rc   3         3         3       18s
root@master:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-7pc2w   1/1     Running   0          94s
nginx-rc-c8d54   1/1     Running   0          94s
nginx-rc-jp8rr   1/1     Running   0          94s
```

→now delete one pod and check pods again created pods automatically:



```
root@master:~# kubectl delete pod nginx-rc-jp8rr
pod "nginx-rc-jp8rr" deleted
```



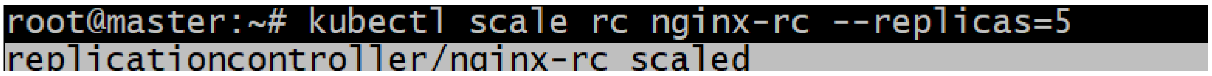
```
root@master:~# kubectl delete pod nginx-rc-jp8rr
pod "nginx-rc-jp8rr" deleted
root@master:~# ^C
root@master:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-7pc2w   1/1     Running   0          22m
nginx-rc-c8d54   1/1     Running   0          22m
nginx-rc-pvcs9   1/1     Running   0          9s
```

4) Scale the ReplicationController from 3 replicas to 5 replicas.

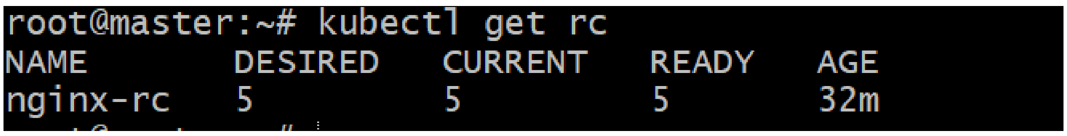
→To scale a ReplicationController from 3 to 5 replicas, we can do it in two ways: using kubectl directly or by modifying the YAML file:

I choose kubectl command:

```
kubectl scale rc nginx-rc --replicas=5
```

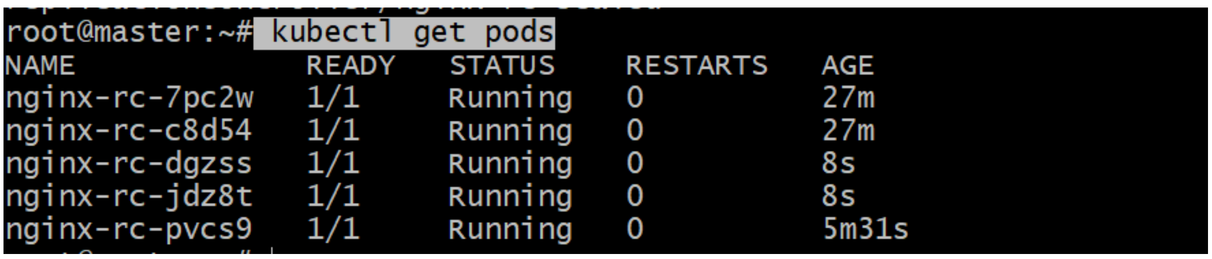


```
root@master:~# kubectl scale rc nginx-rc --replicas=5
replicationcontroller/nginx-rc scaled
```



```
root@master:~# kubectl get rc
NAME       DESIRED   CURRENT   READY   AGE
nginx-rc   5         5         5       32m
```

→check pods..2 more created:

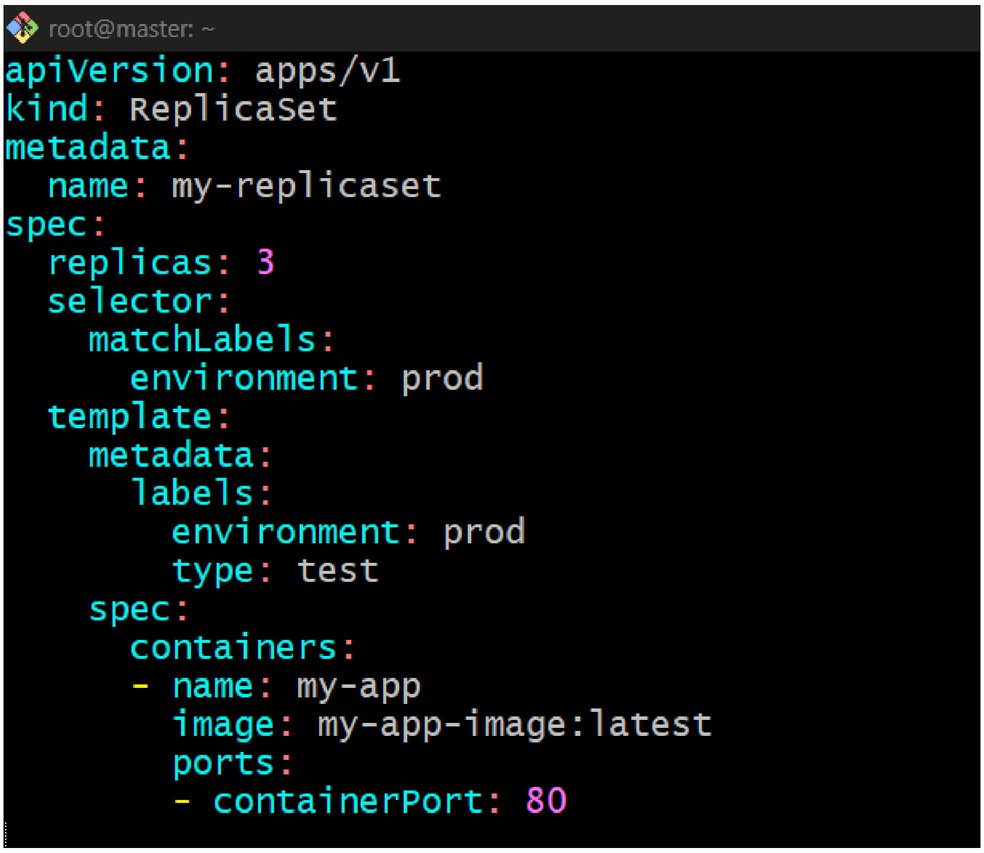


```
root@master:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-7pc2w   1/1     Running   0          27m
nginx-rc-c8d54   1/1     Running   0          27m
nginx-rc-dgzss   1/1     Running   0          8s
nginx-rc-jdz8t   1/1     Running   0          8s
nginx-rc-pvcs9   1/1     Running   0          5m31s
```

5) Create a ReplicaSet to manage pods based on multiple labels (prod and test).

→ a ReplicaSet to manage pods based on multiple labels (prod and test):

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: prod
  template:
    metadata:
      labels:
        environment: prod
        type: test
    spec:
      containers:
      - name: my-app
        image: my-app-image:latest
        ports:
        - containerPort: 80
```

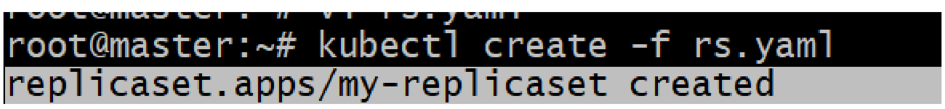


```
root@master: ~
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: my-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      environment: prod
  template:
    metadata:
      labels:
        environment: prod
        type: test
    spec:
      containers:
      - name: my-app
        image: my-app-image:latest
        ports:
        - containerPort: 80
```

→**run the yaml file:**

kubectl create -f rs.yaml

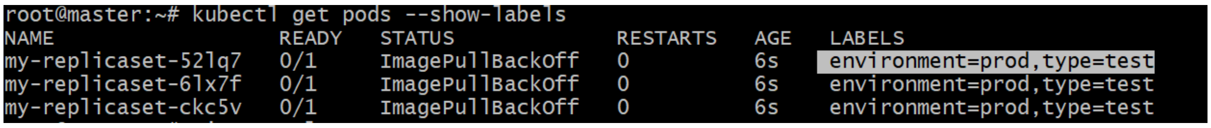
created rs



```
root@master:~# kubectl create -f rs.yaml
replicaset.apps/my-replicaset created
```

→**ckeck labels:**

kubectl get pods --show-labels



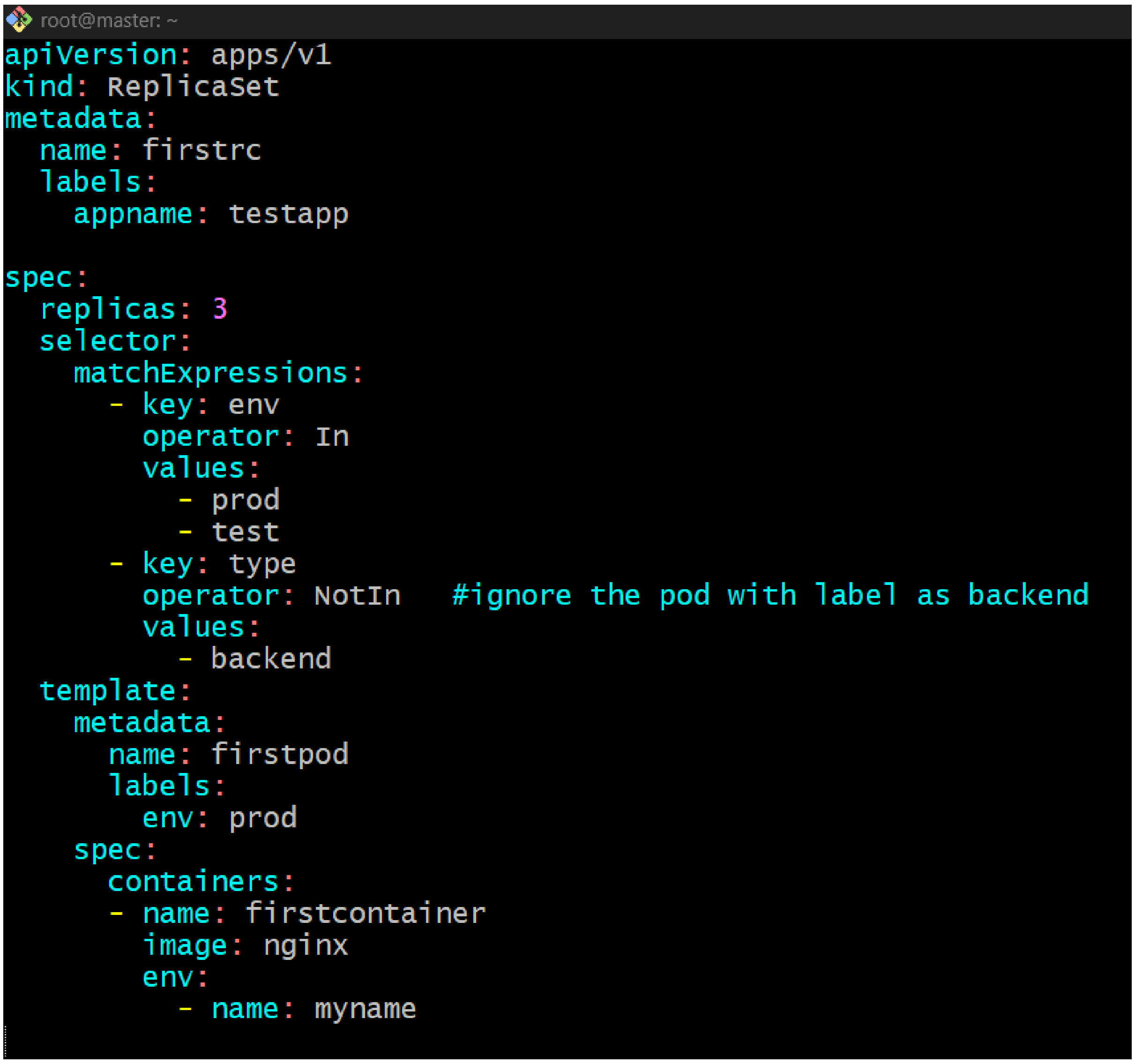
```
root@master:~# kubectl get pods --show-labels
NAME                  READY   STATUS             RESTARTS   AGE   LABELS
my-replicaset-52lq7   0/1     ImagePullBackOff   0          6s    environment=prod,type=test
my-replicaset-6lx7f   0/1     ImagePullBackOff   0          6s    environment=prod,type=test
my-replicaset-ckc5v   0/1     ImagePullBackOff   0          6s    environment=prod,type=test
```

6) Deploy a ReplicaSet that excludes pods with the label backend.

→ ReplicaSet that excludes pods with the label backend:

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: firstrc
  labels:
    appname: testapp

spec:
  replicas: 3
  selector:
    matchExpressions:
      - key: env
        operator: In
        values:
          - prod
          - test
      - key: type
        operator: NotIn   #ignore the pod with label as backend
        values:
          - backend
  template:
    metadata:
      name: firstpod
      labels:
        env: prod
    spec:
      containers:
      - name: firstcontainer
        image: nginx
        env:
          - name: myname
```

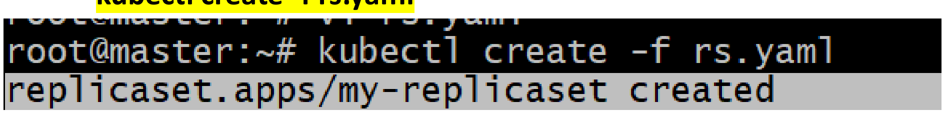


```
root@master: ~
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: firstrc
  labels:
    appname: testapp

spec:
  replicas: 3
  selector:
    matchExpressions:
      - key: env
        operator: In
        values:
          - prod
          - test
      - key: type
        operator: NotIn   #ignore the pod with label as backend
        values:
          - backend
  template:
    metadata:
      name: firstpod
      labels:
        env: prod
    spec:
      containers:
      - name: firstcontainer
        image: nginx
        env:
          - name: myname
```

→run yaml file:

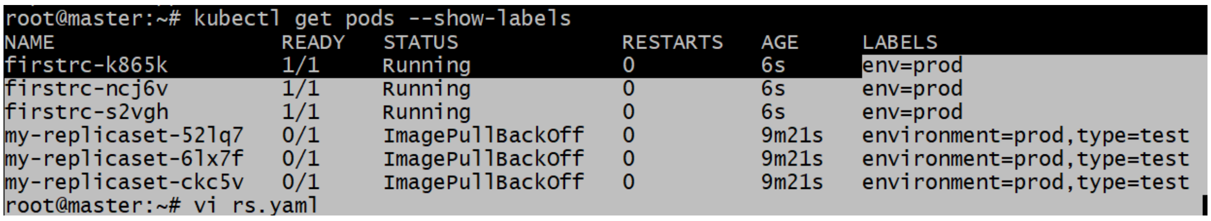
kubectl create -f rs.yaml



```
root@master:~# kubectl create -f rs.yaml
replicaset.apps/my-replicaset created
```

→check labels:

--here It ignores backend type



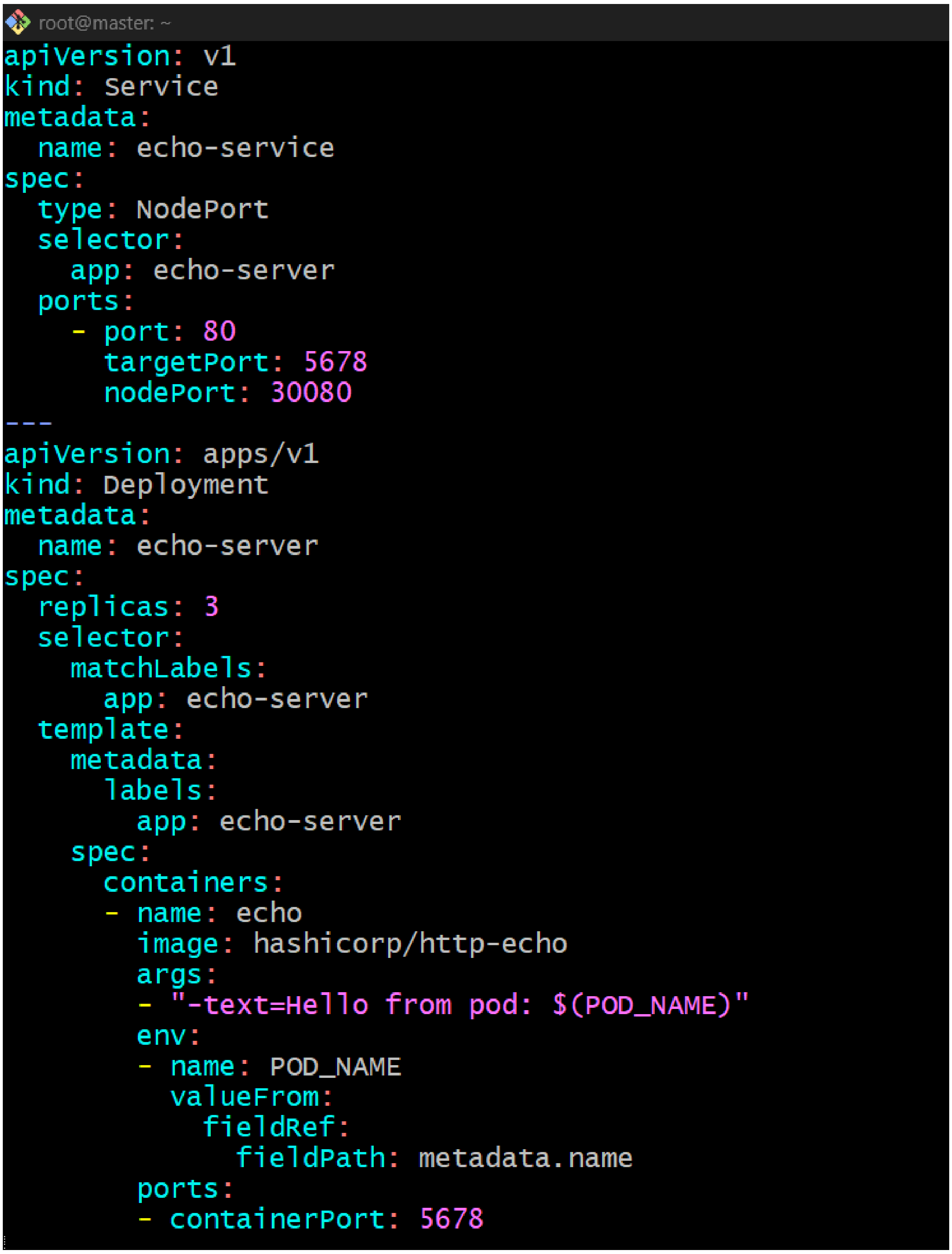
```
root@master:~# kubectl get pods --show-labels
NAME                  READY   STATUS             RESTARTS   AGE     LABELS
firstrc-k865k         1/1     Running            0          6s      env=prod
firstrc-ncj6v         1/1     Running            0          6s      env=prod
firstrc-s2vgh         1/1     Running            0          6s      env=prod
my-replicaset-52lq7   0/1     ImagePullBackOff   0          9m21s   environment=prod,type=test
my-replicaset-6lx7f   0/1     ImagePullBackOff   0          9m21s   environment=prod,type=test
my-replicaset-ckc5v   0/1     ImagePullBackOff   0          9m21s   environment=prod,type=test
root@master:~# vi rs.yaml
```

7) Test load balancing across multiple pods using a NodePort service.

→yaml file to create multiple pods and create a nodeport service:

```
apiVersion: v1
kind: Service
metadata:
  name: echo-service
spec:
  type: NodePort
  selector:
    app: echo-server
  ports:
    - port: 80
      targetPort: 5678
      nodePort: 30080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-server
spec:
  replicas: 3
  selector:
    matchLabels:
      app: echo-server
  template:
    metadata:
      labels:
        app: echo-server
```

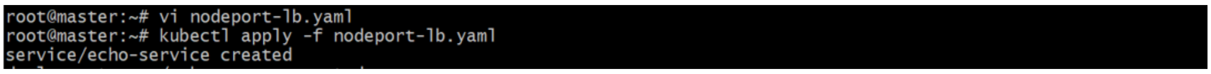
```
spec:
  containers:
  - name: echo
    image: hashicorp/http-echo
    args:
    - "-text=Hello from pod: $(POD_NAME)"
    env:
    - name: POD_NAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
    ports:
    - containerPort: 5678
```



```
root@master: ~
apiVersion: v1
kind: Service
metadata:
  name: echo-service
spec:
  type: NodePort
  selector:
    app: echo-server
  ports:
    - port: 80
      targetPort: 5678
      nodePort: 30080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: echo-server
spec:
  replicas: 3
  selector:
    matchLabels:
      app: echo-server
  template:
    metadata:
      labels:
        app: echo-server
    spec:
      containers:
      - name: echo
        image: hashicorp/http-echo
        args:
        - "-text=Hello from pod: $(POD_NAME)"
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        ports:
        - containerPort: 5678
```

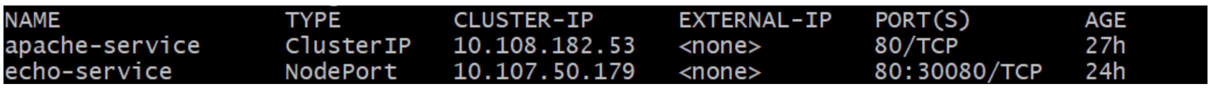
→run the yaml:

kubectl apply -f nodeport-lb.yaml



```
root@master:~# vi nodeport-lb.yaml
root@master:~# kubectl apply -f nodeport-lb.yaml
service/echo-service created
```

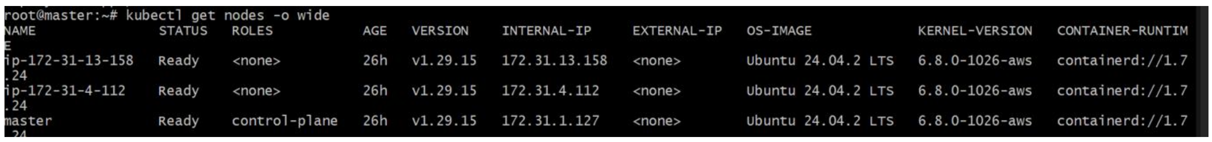
→created service:



```
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)         AGE
apache-service    ClusterIP   10.108.182.53   <none>        80/TCP          27h
echo-service      NodePort    10.107.50.179   <none>        80:30080/TCP    24h
```

→apply this command to get a node ip:

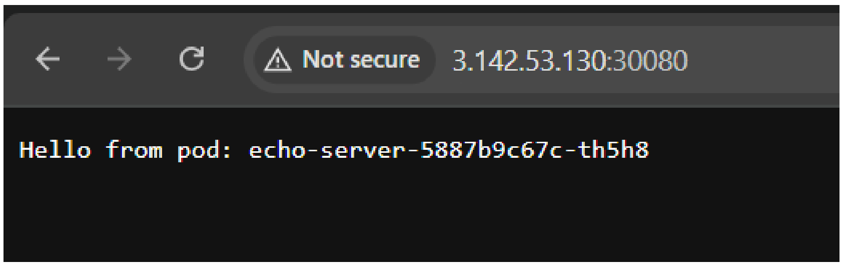
kubectl get nodes -o wide



```
root@master:~# kubectl get nodes -o wide
NAME               STATUS   ROLES           AGE   VERSION    INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION    CONTAINER-RUNTIM
E
ip-172-31-13-158   Ready    <none>          26h   v1.29.15   172.31.13.158   <none>        Ubuntu 24.04.2 LTS   6.8.0-1026-aws    containerd://1.7
.24
ip-172-31-4-112    Ready    <none>          26h   v1.29.15   172.31.4.112    <none>        Ubuntu 24.04.2 LTS   6.8.0-1026-aws    containerd://1.7
.24
master             Ready    control-plane   26h   v1.29.15   172.31.1.127    <none>        Ubuntu 24.04.2 LTS   6.8.0-1026-aws    containerd://1.7
```

→now access it on browser worker-01 pub ip:nodeport:

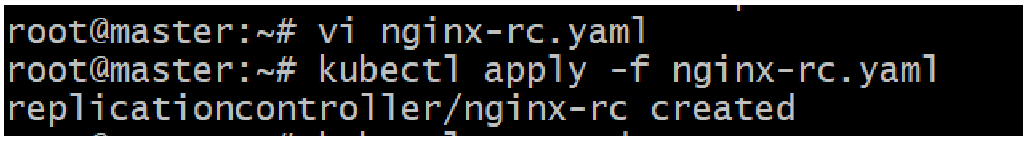
We can see load balancing in action….



```
Hello from pod: echo-server-5887b9c67c-th5h8
```

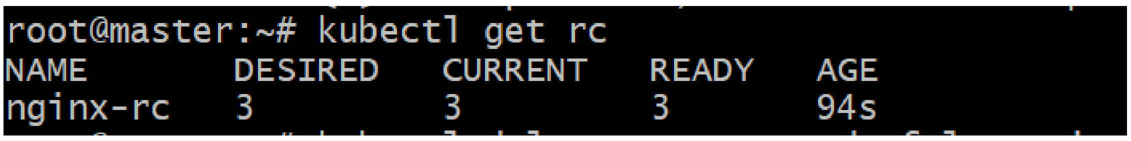
8) Delete a ReplicationController without affecting the running pods

→ first create a replica set:



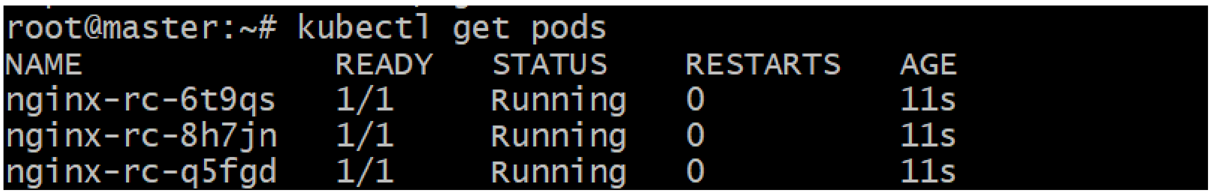
```
root@master:~# vi nginx-rc.yaml
root@master:~# kubectl apply -f nginx-rc.yaml
replicationcontroller/nginx-rc created
```

→check rc:



```
root@master:~# kubectl get rc
NAME       DESIRED   CURRENT   READY   AGE
nginx-rc   3         3         3       94s
```

→check pods:

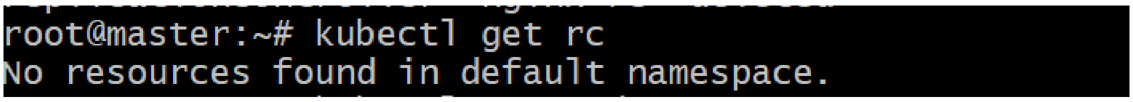


```
root@master:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-6t9qs   1/1     Running   0          11s
nginx-rc-8h7jn   1/1     Running   0          11s
nginx-rc-q5fgd   1/1     Running   0          11s
```

→now Delete a ReplicationController without affecting the running pods using below command:

kubectl delete rc --cascade=false nginx-rc

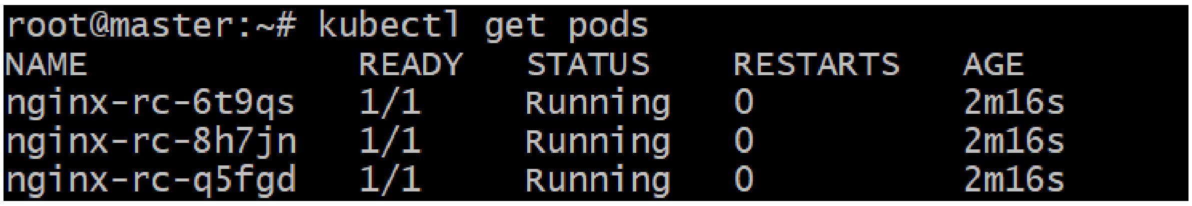
now check Rc



```
root@master:~# kubectl get rc
No resources found in default namespace.
```

Deleted rc

Now check pods



```
root@master:~# kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
nginx-rc-6t9qs    1/1     Running   0          2m16s
nginx-rc-8h7jn    1/1     Running   0          2m16s
nginx-rc-q5fgd    1/1     Running   0          2m16s
```

Pods are not deleted…..