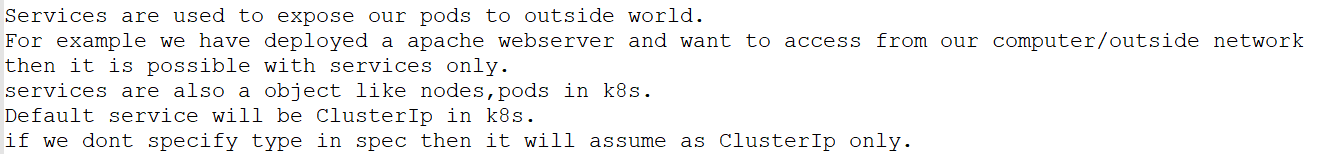
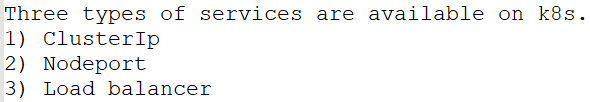
**Kubernetes-03**

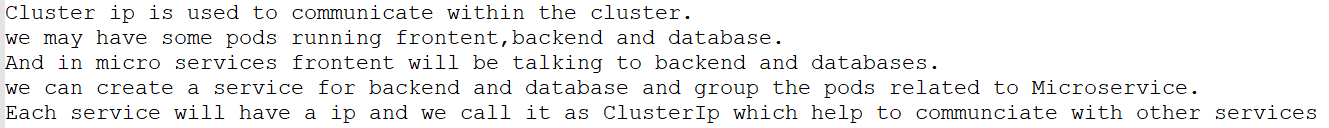
**SERVICES:**

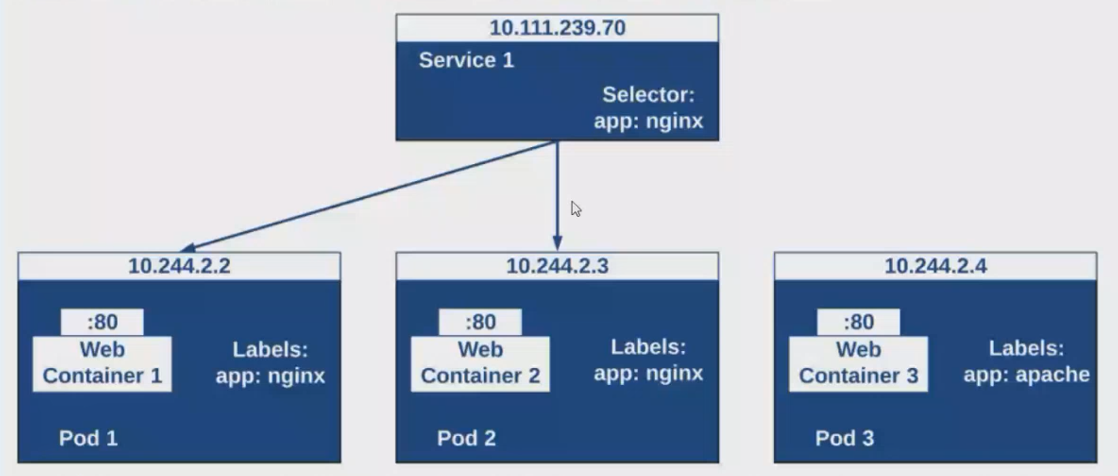
****

**TYPES OF SERVICES:**

****

**CLUSTERIP:**

****

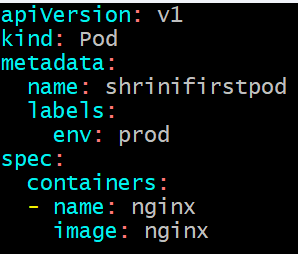
****

**SERVICES are also help in load balancer   
if we have three pods are with same containers are running in three different nodes machine then whatever request assign to the services then it will redirect to any of the node machine and, if labels are same as shown in above pics**

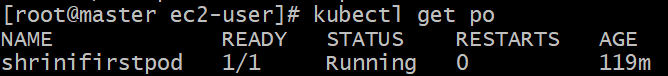
**Services will be mapped based upon the labels**

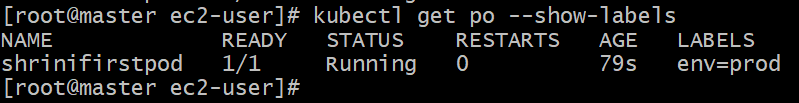
**Example:**

**Here we are creating pod with the labels,**

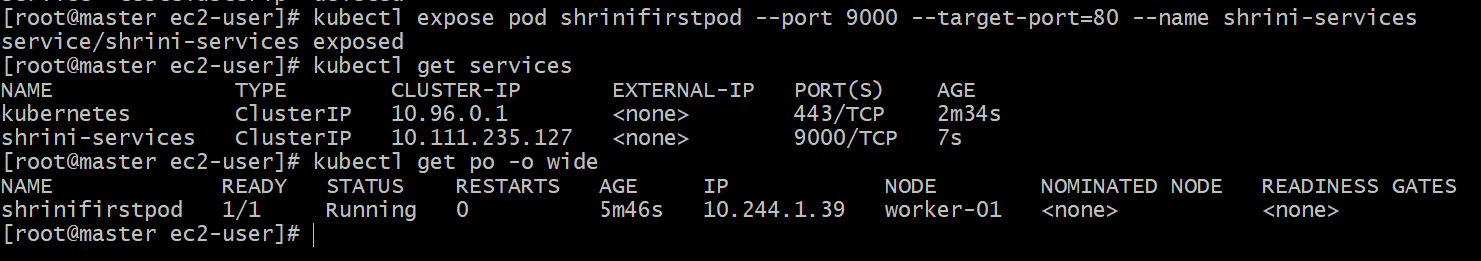
**Vi shrinipod.yaml  
**

****

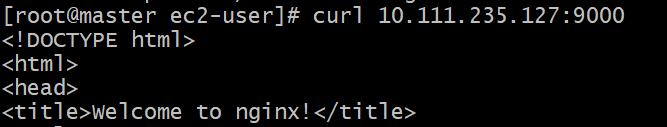
****

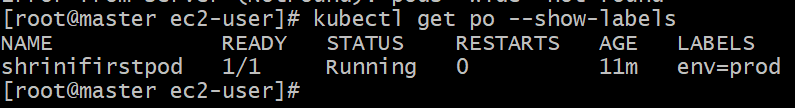
**To check the labels of your current pods:  
**

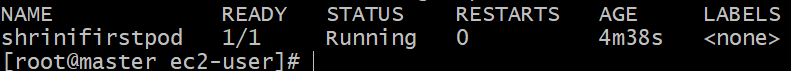
**Now we will expose our pod-container with some ports:**

****

**Then I will try with curl command to check its working or not but here you can see that with the services of clusterIP has different IP and its get connected with our pod-container through labels**

****

****

**And now I will remove the labels from the existing pod:  
**

**And now if I tried to do curl command without any label then:**

****

**So final conclusion is:-  
Based upon the labels name only the service will be assigned to the pod and that’s the reason we are using labels in both pods and services**

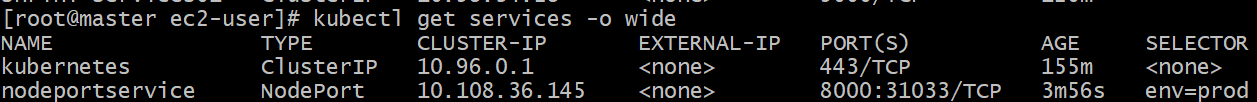
**If we want to expose the service outside the world on that case we have another service “NODE PORT”**

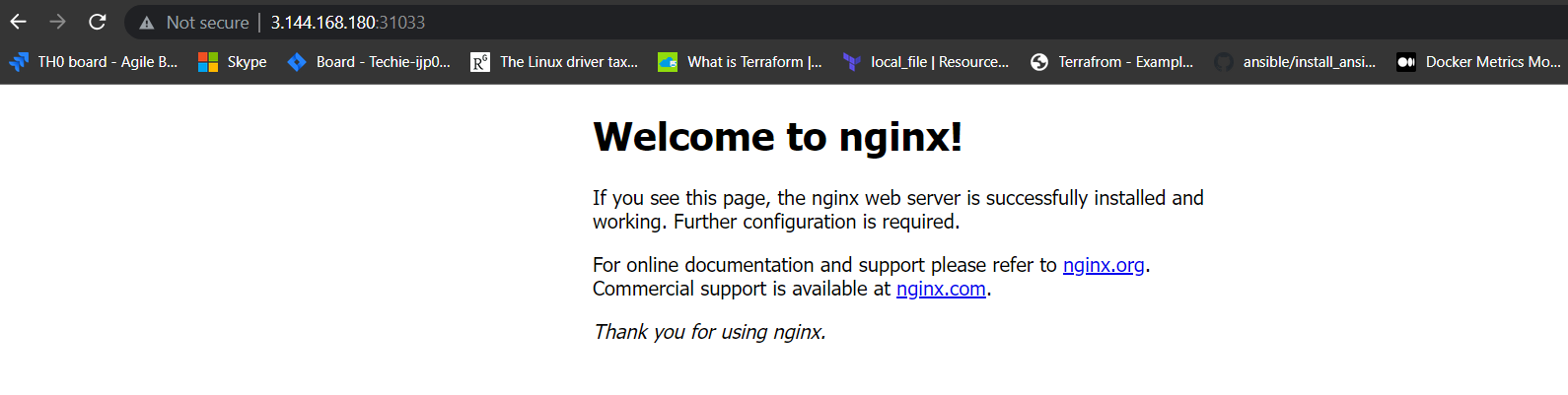
**Node port:**

****

**How to create Node Port?**

****

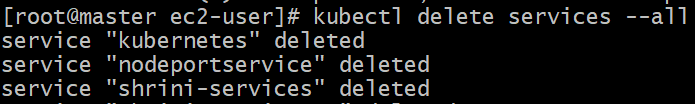
****

**And, now if we want to check the our services then we will take public ip of our master and the node port  
**

**Port can be assigned between (30000 - 32767)**

**It will in both worker and master public ip it means Any request which is coming to the node-port then it can be  
accessed**

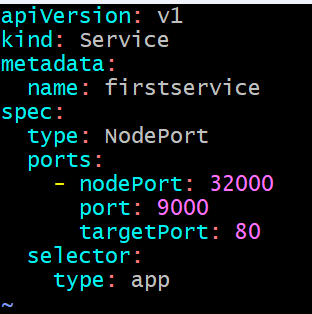
**Now, how to delete any service:**

****

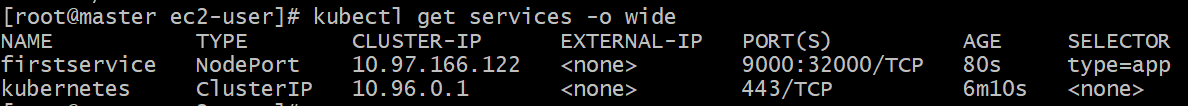
**How services will work?**

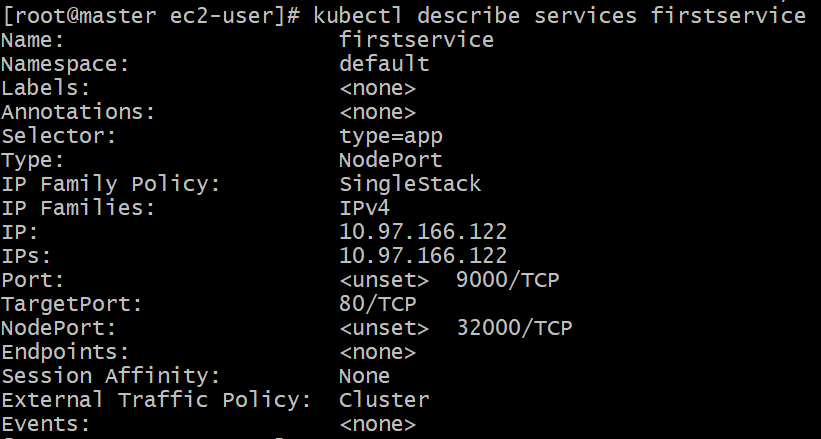
**Services will check the request which is sent on port and redirect that based on label.**

**First we will create service.yml**

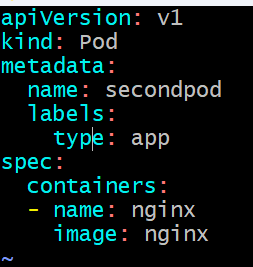
****

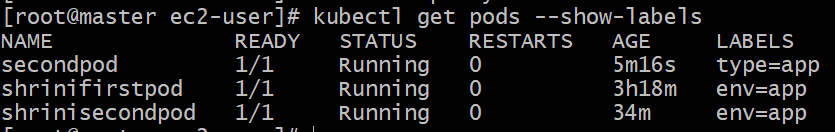
****

****

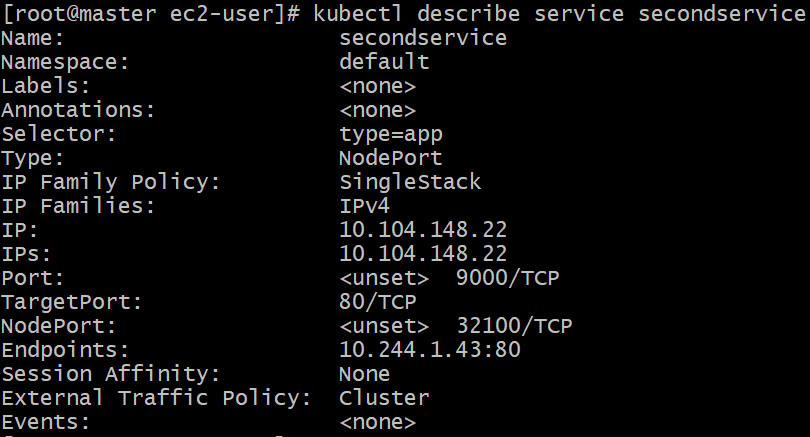
**How to check the details of our services;  
**

**So, here we are mentioned selectors (type=app) which we can use it on pod yaml;**

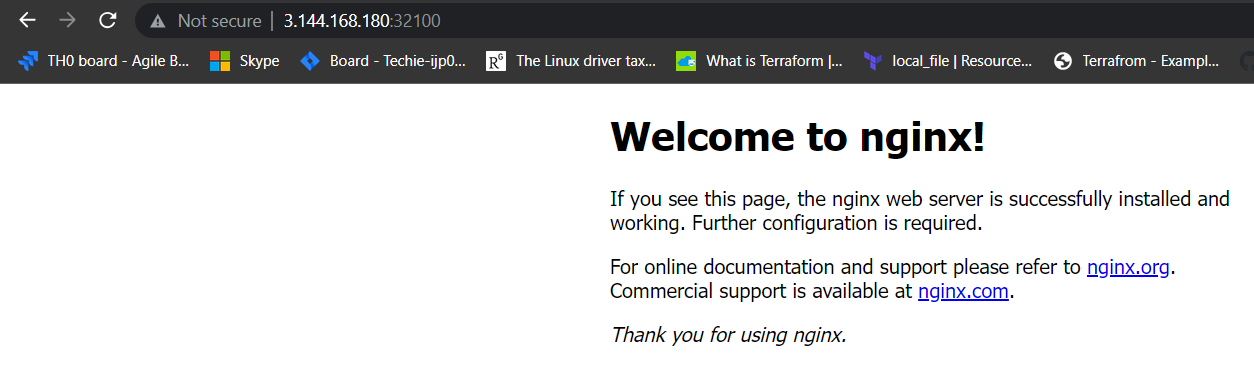
****

****

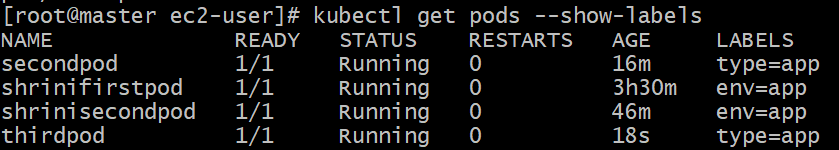
**Now our pod has been configured with the services with the help of labels:**

****

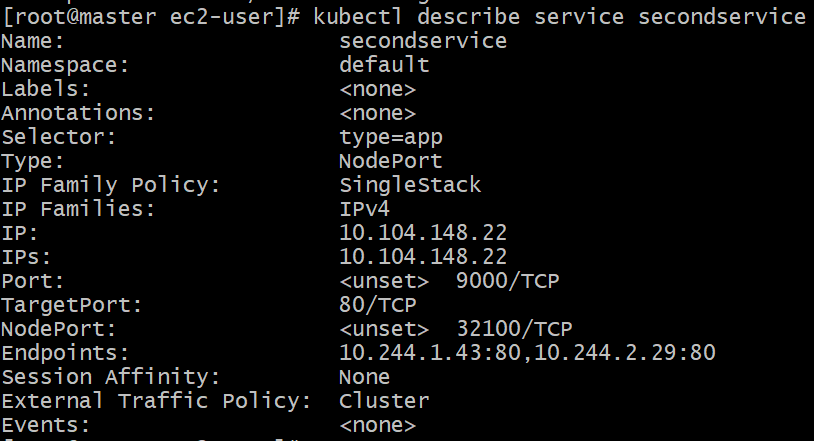
**Here we get endpoint by that end point we can access our service within the cluster and with the public ip we can access through node port of “32100”**

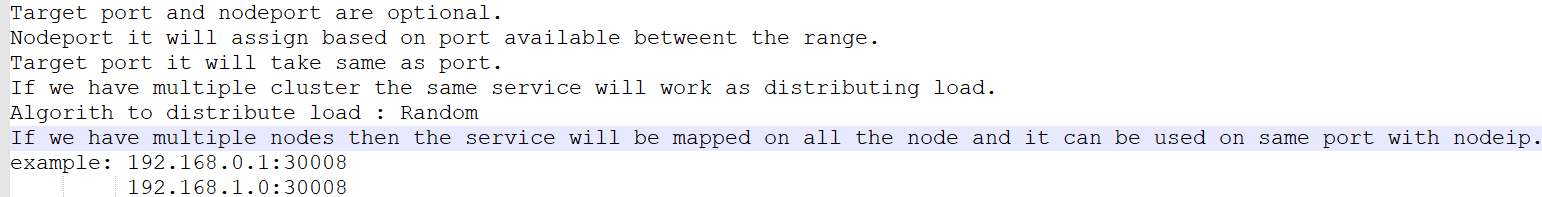
****

**And with the help of label we can assign the services with multiple pods:**

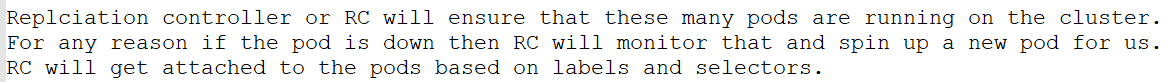
****

**And now we have two endpoints for the same service with the help of label:**

****

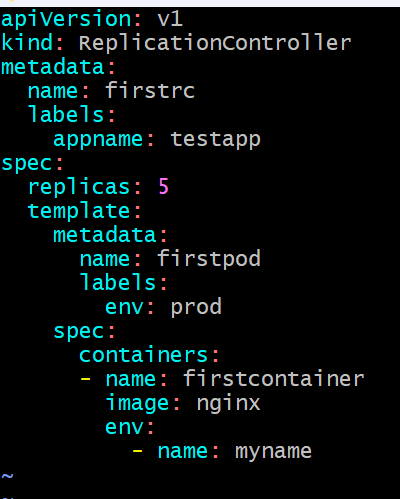
****

**Replication Controller:**

****

**How to create a RC?**

**Vi ReplicationController**

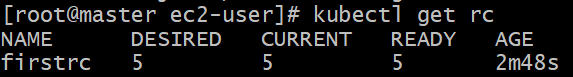
****

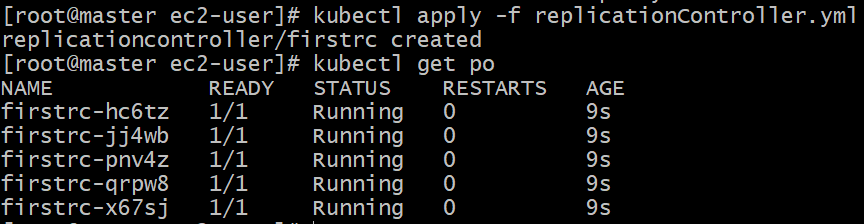
**In metadata we are giving name of our RC and with the label of “appname: testapp”**

**Then in the spec we are mentioned 5 pods with the name of our pod which we want to get replicas**

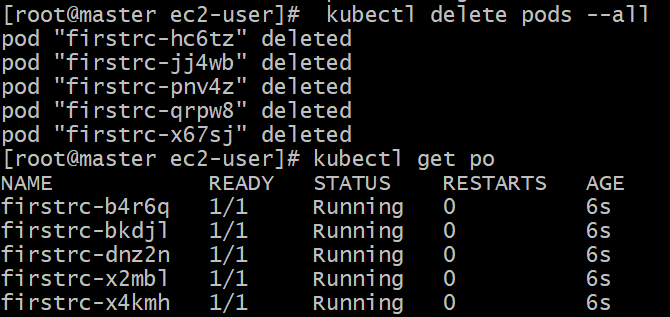
**And under that we are mention about containers of our pod,**

**Then we will create our replication controller and the pod will take the name which we mentioned on the replicationcontroller.yml**

****

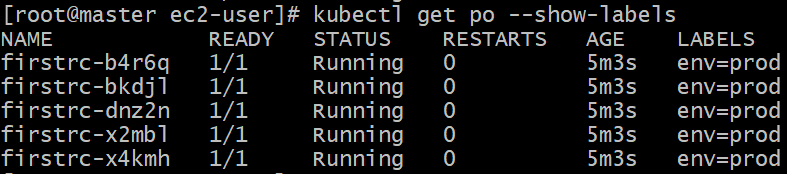
****

**And now if we delete any pod then it automatically take up the pods which are under the replication controller;**

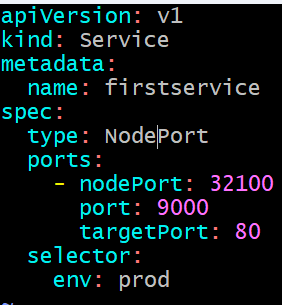
****

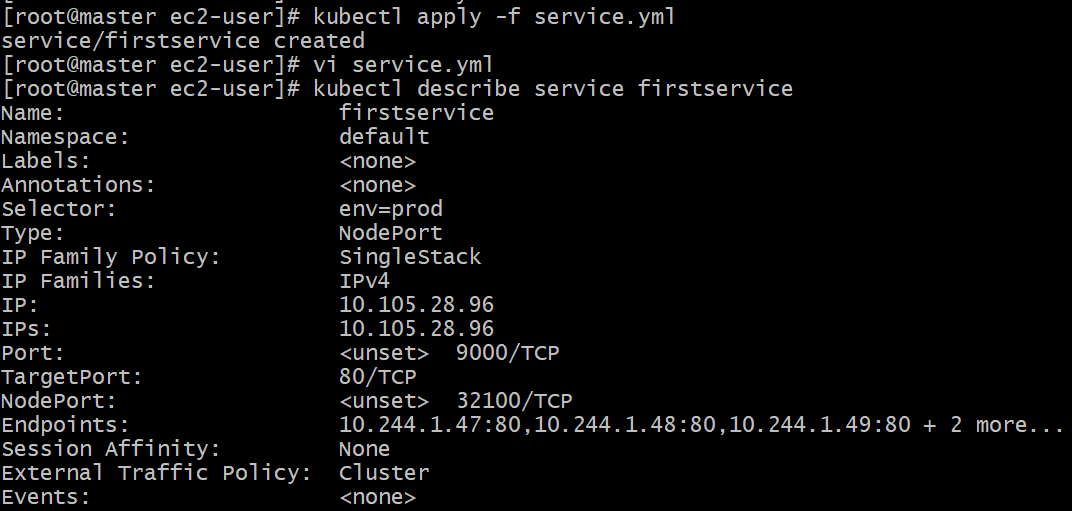
**It will always maintains the value of pods which we mentioned on the rc.yaml**

**And the labels for the rc-pods are also same because we mentioned on the yaml**

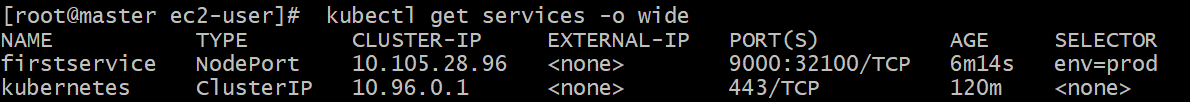
****

**Now we will create a service with the label which we mentioned on the rc.yml**

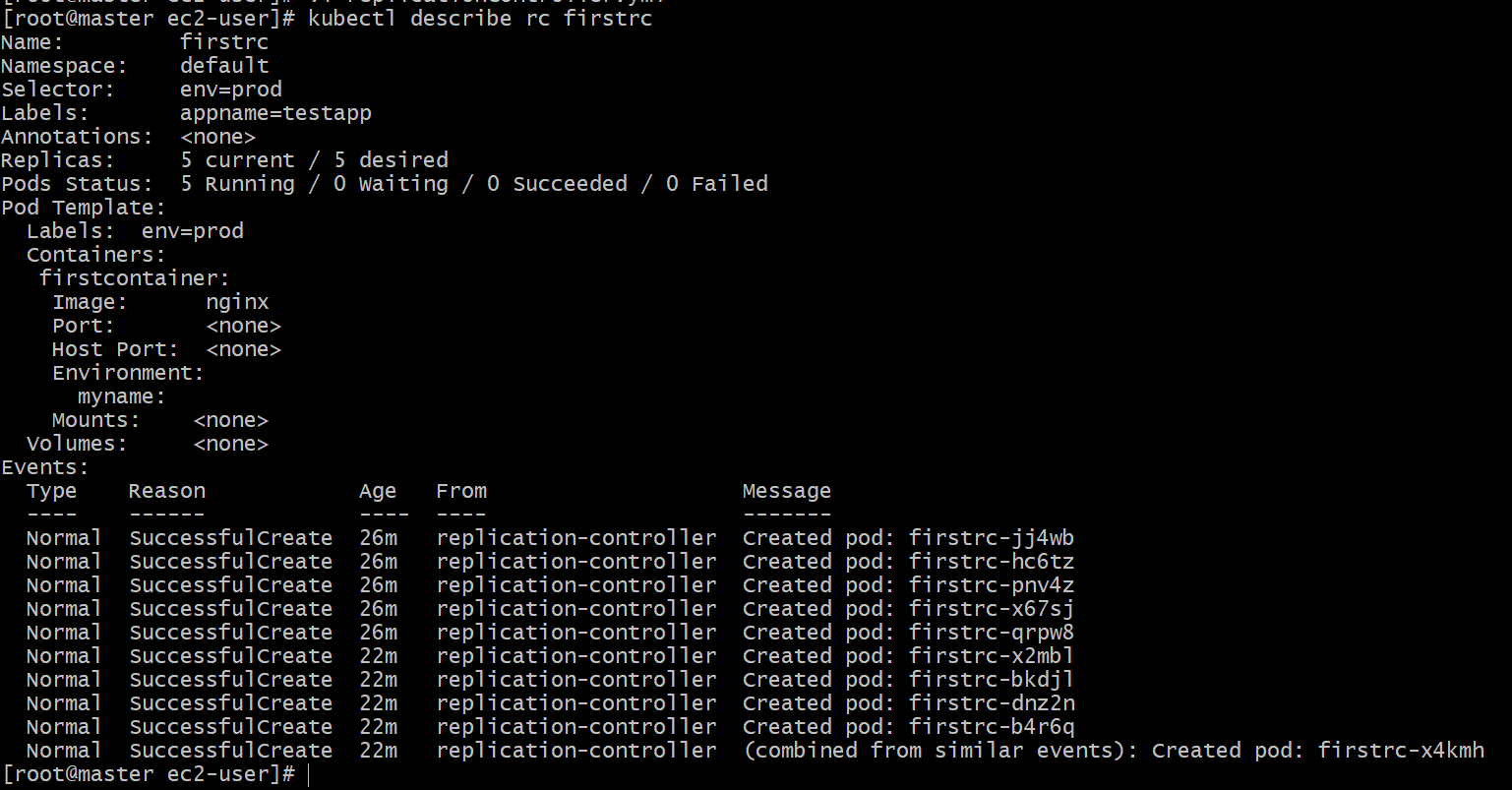
****

****

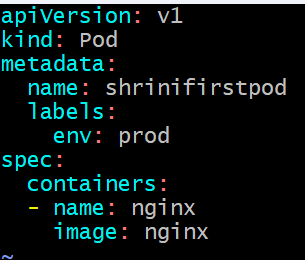
**Here we are getting five endpoints because with that label we have 5pods so it is giving five endpoints,**

****

**To get the Replication controller info we can use:**

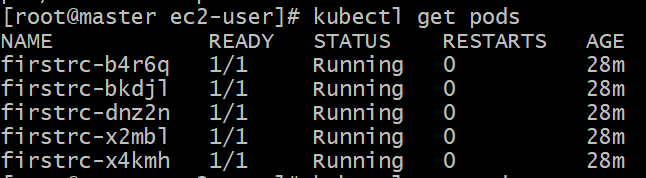
****

**And now I m giving a pod to same label of rc then:**

****

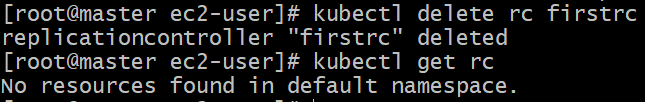
****

**Pod is created but it is not showing in the list of pods:**

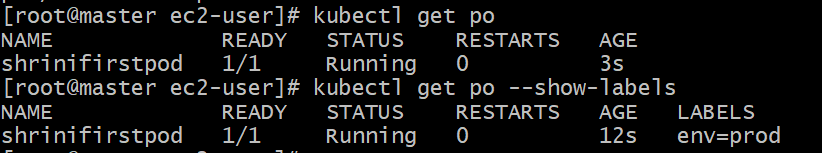
****

**Because, in the replication controller.yml we mentioned 5 replicas there should be worked on that labels**

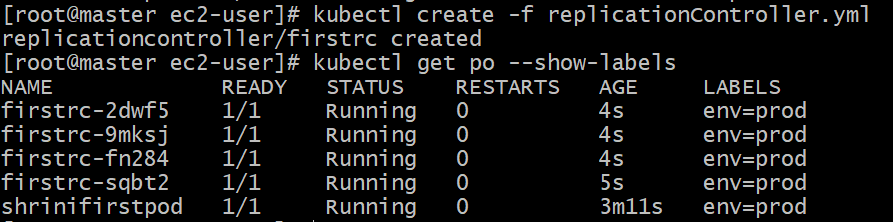
**So, now if we want to delete the replication controller we can use:-**

****

**And the pod which we tried in above steps it will get created with that label**

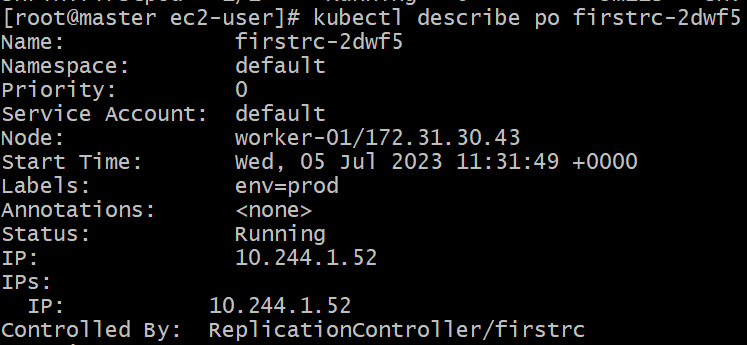
****

**In the next we once again create replica with the yaml and replicas will mentioned 5 then, it will create only “4”**

****

**Because in the definition of rc we mentioned that it works on the selectors and label if it is fulfilled then it won’t create any pods**

**And we can find with the describe command that this pod is controlled by “ReplicationController”**

****

**And the pods which are not created by ReplicationController then it will not show any above thing like “controlled By:rc”**

**If two pods are already running and we have executed our rc with 7 and there is no owner for that pod then it will get the owner like “controlled By: replication controller”**

**And if we create any service file with the same selector which we used under the labels of replication controller.yaml   
then it get Automatically redirected to that service**

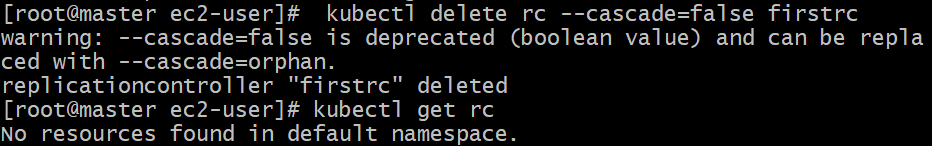
**Why do we need Labels and selectors?**

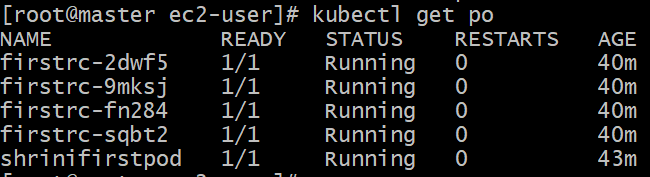
* **If suppose we have 100 of containers running then labels and selectors will help**
* **RC to filter the containers and apply monitoring based on labels and selectors.**

**Will RC can be configured to the existing running pods?**

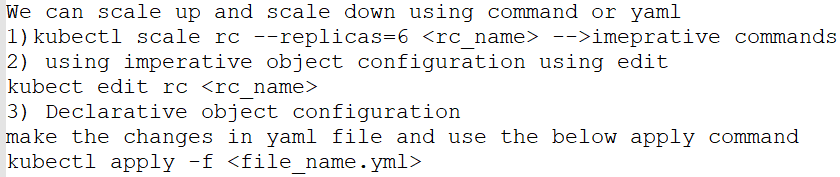
**Yes, we can configure RS to running containers by defining the selectors in RC.yml**

**How to delete on rc and the pods should be running?**

****

****

**How to scale up/scale down using Rc?**

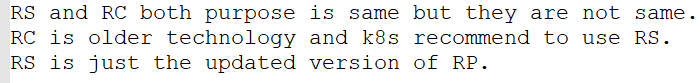
****

**SELECTORS AND LABELS IN RC USE-CASE;**

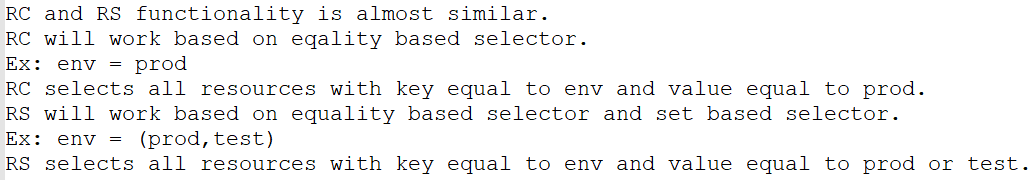
****

**------------------------------------------------------------------------------------**

**Replica set:**

****

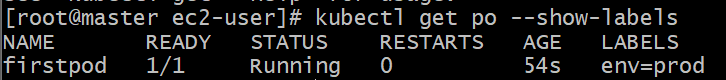
**Replication controller vs Replica set Difference?**

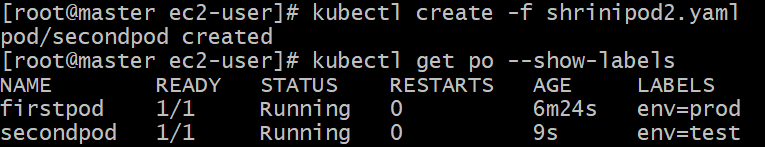
****

**We have two types of selectors:**

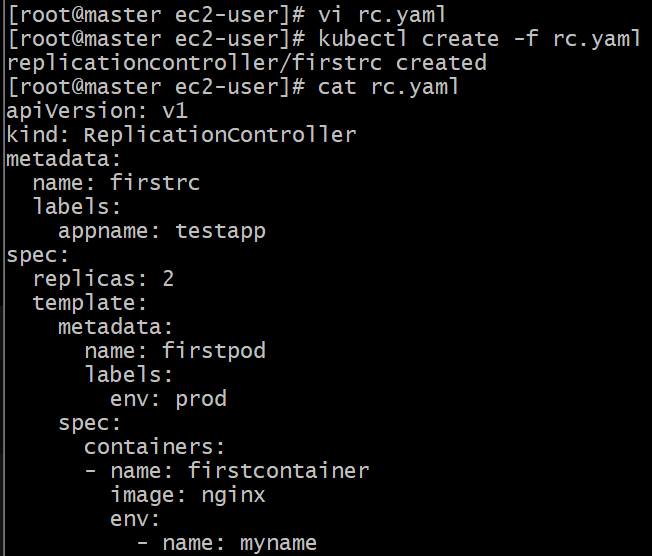
**1. set-based selector 🡪 RS  
2. Equality-based selector 🡪 RC**

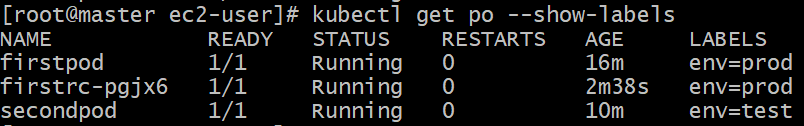
**Explanation:  
First we creating a pod with the create command;  
**

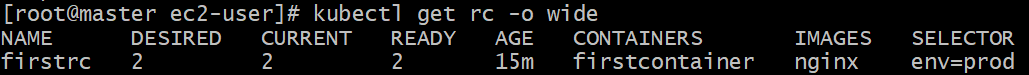
**Then we check the labels:  
**

**And now we are creating second pod with different label;  
**

**One RC we will create by the rc.yaml file**

****

****

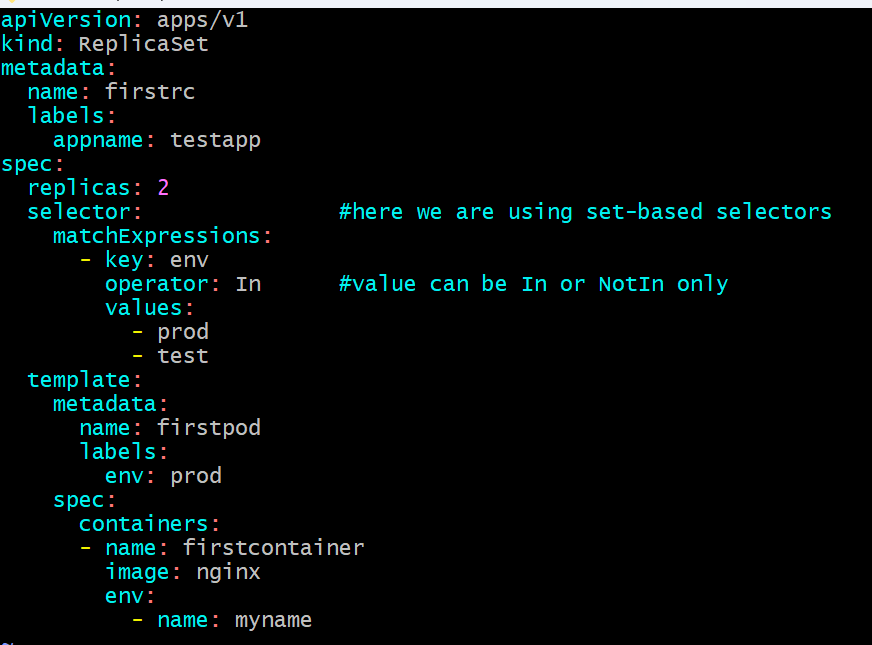
****

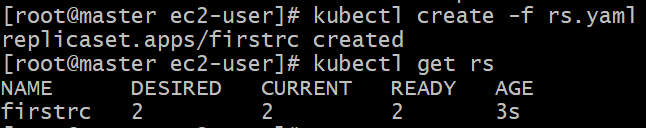
**On the above yaml file of “rc” we mentioned the replica with “2” so, In RC it selects all resources with key equal to env and value equal to prod. [Env=prod]**

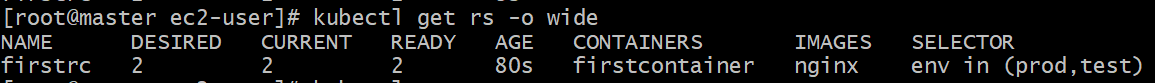
**But in Replication set it can selects all resources with key equal to env and value equal to prod or test. [Env=prod, test]**

**So that’s why we used “Replication Sets”**

**Now we will create one replication sets with rs.yaml file:**

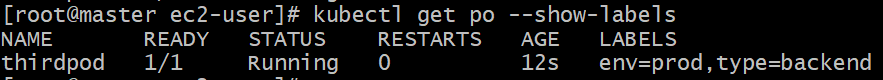
****

****

**So, now it will create 2 pods under the value of rs.yaml with  
the set-base selector (prod, test)**

**Ignoring scenario:-**

**If Suppose we have 3 pod now with label one as"prod","test" and one pod with two labels "prod" and "backend" and we want to ignore a specific pods then how can we use this rs.yaml:-**

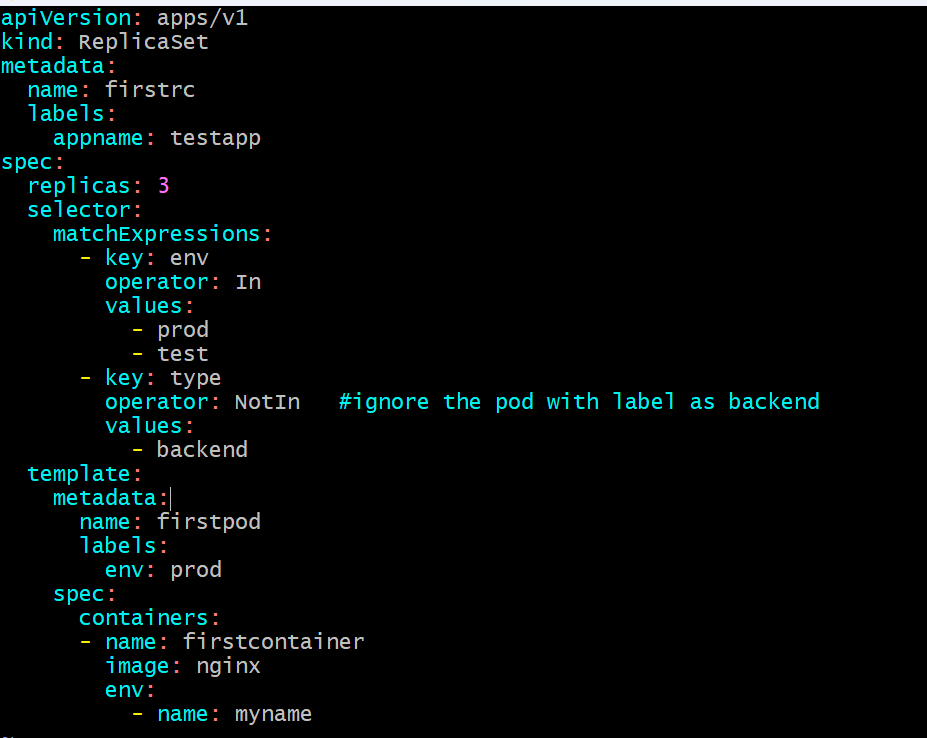
****

**So with the help of “backend” we can manage our labels for that we have given below yaml:**

**First we will delete the yaml and then create a new one with the new content of backend and here we will use;**

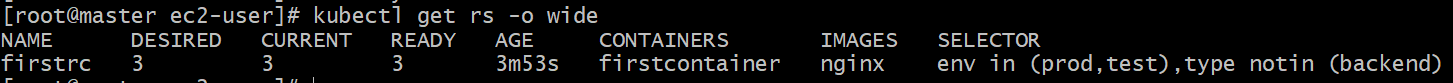
**“NotIn 🡪 operator”**

**It will “#ignore the pod with label as backend”**

**Vi rs.yaml:  
**

****

**Now it will ignore the backend one and then it will create the three pods by the help of “NotIn” operator;**

****

**--------------------------------Shrinivas----------------------------------------**