1. **Why do we use Kubernetes? what is a container orchestration tool?**

Answer: container orchestration tool which is used to mange the containers by automating the various concerns around container “running.”

Example of container orchestration tool-

Kubernetes

OpenShift

Docker Swarm

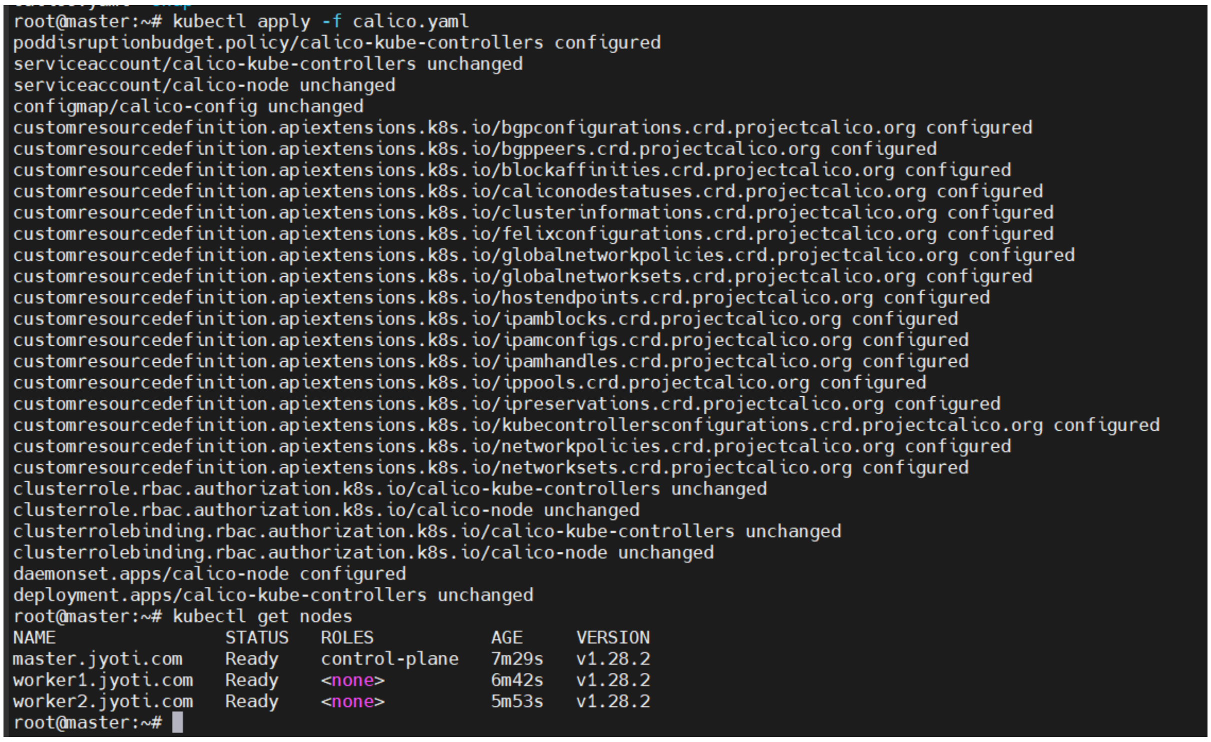
We use Kubernetes because of these features-

- Kubernetes is a open-source tool
- Automation of deployment and scalability
- It provides load balancing
- it has a feature of self healing
- it makes the application highly available
- it uses the resources efficiently
- you can make n number of container through on command.
- Kubernetes works on both vertical and horizontal scaling.
- Works on client server model
- Kubernetes support

2. **What were the challenges we faced in docker? Mention the answer in bullet points**
    1. There has no Autoscaling feature
    2. There has no Load balancing feature
    3. There has no Self – healing feature
    4. you can not make multiple container through on command. You will have to run command manually for each container.
    5. Docker prefer vertical scaling.

3. **What components are used in Kubernetes architecture and what model do we use to manage Kubernetes on our system?**
   Create Kubernetes cluster (take whichever machine you like atleast use 1 master node and 2 worker nodes)
   Answer- In Kubernetes architecture there is cluster which is a combination of master node and worker node.
   Master node/control panel component-
    - kube-apiserver
    - etcd
    - kube-scheduler
    - kube-controller-manager

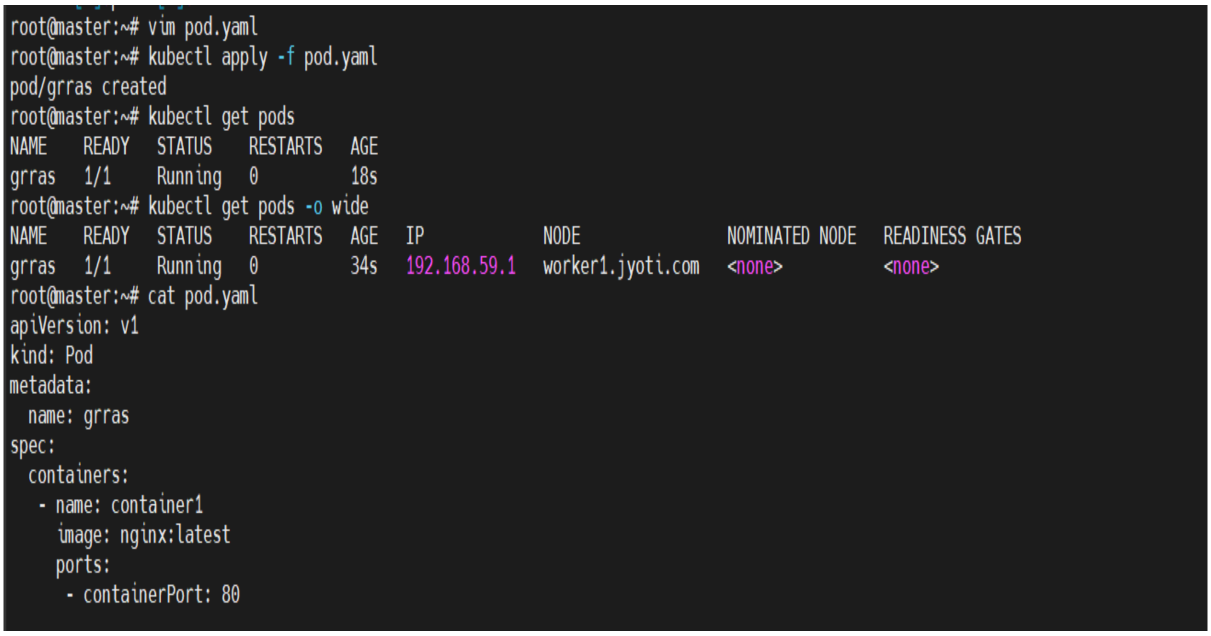
   Worker node component-
    - kubelet
    - kube-proxy
    - Container runtime



```
root@master:~# kubectl apply -f calico.yaml
poddisruptionbudget.policy/calico-kube-controllers configured
serviceaccount/calico-kube-controllers unchanged
serviceaccount/calico-node unchanged
configmap/calico-config unchanged
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org configured
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org configured
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers unchanged
clusterrole.rbac.authorization.k8s.io/calico-node unchanged
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers unchanged
clusterrolebinding.rbac.authorization.k8s.io/calico-node unchanged
daemonset.apps/calico-node configured
deployment.apps/calico-kube-controllers unchanged
root@master:~# kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
master.jyoti.com    Ready    control-plane   7m29s   v1.28.2
worker1.jyoti.com   Ready    <none>          6m42s   v1.28.2
worker2.jyoti.com   Ready    <none>          5m53s   v1.28.2
root@master:~#
```

4. **What is a Pod? create a pod name grras which uses the image of nginx with port 80 allowed in the manifest file.**

Answer- A pod is the smallest execution unit which is managed by Kubernetes.



```
root@master:~# vim pod.yaml
root@master:~# kubectl apply -f pod.yaml
pod/grras created
root@master:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
grras   1/1     Running   0          18s
root@master:~# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP             NODE                NOMINATED NODE   READINESS GATES
grras   1/1     Running   0          34s   192.168.59.1   worker1.jyoti.com   <none>           <none>
root@master:~# cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: grras
spec:
  containers:
   - name: container1
     image: nginx:latest
     ports:
      - containerPort: 80
```

5. **What are the strategies we use to achieve HA (high availability)for an application? (hint: Deployment strategies).**

   Answer:

   1. **Replication Controller**

      A *Replication Controller* ensures that a specified number of pod replicas are running at any one time.

      Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.

   2. **Replica set**

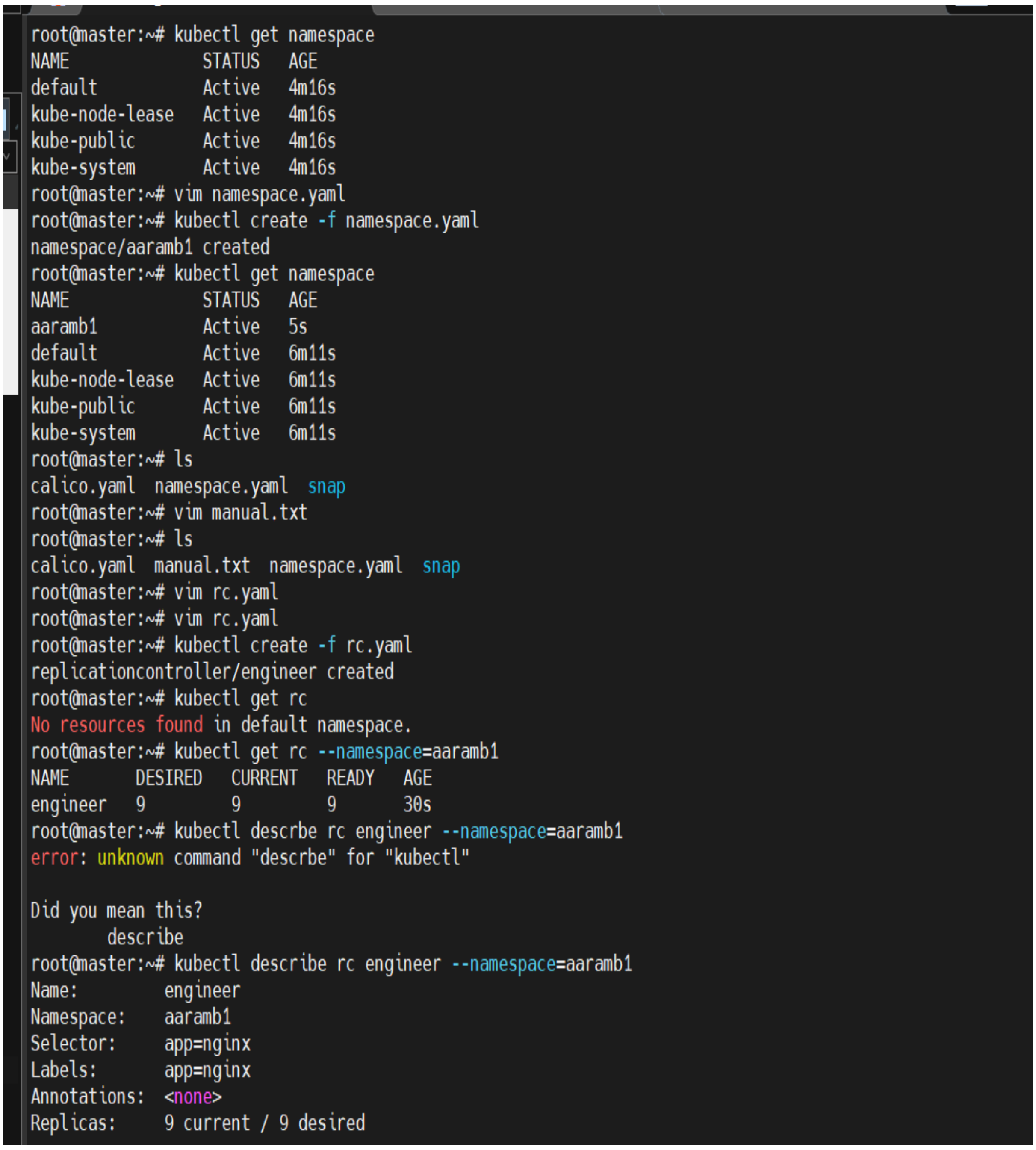
      A Replica Set's purpose is to maintain a stable set of replica Pods running at any given time.

      it is often used to guarantee the availability of a specified number of identical Pods

   3. **Deployment**

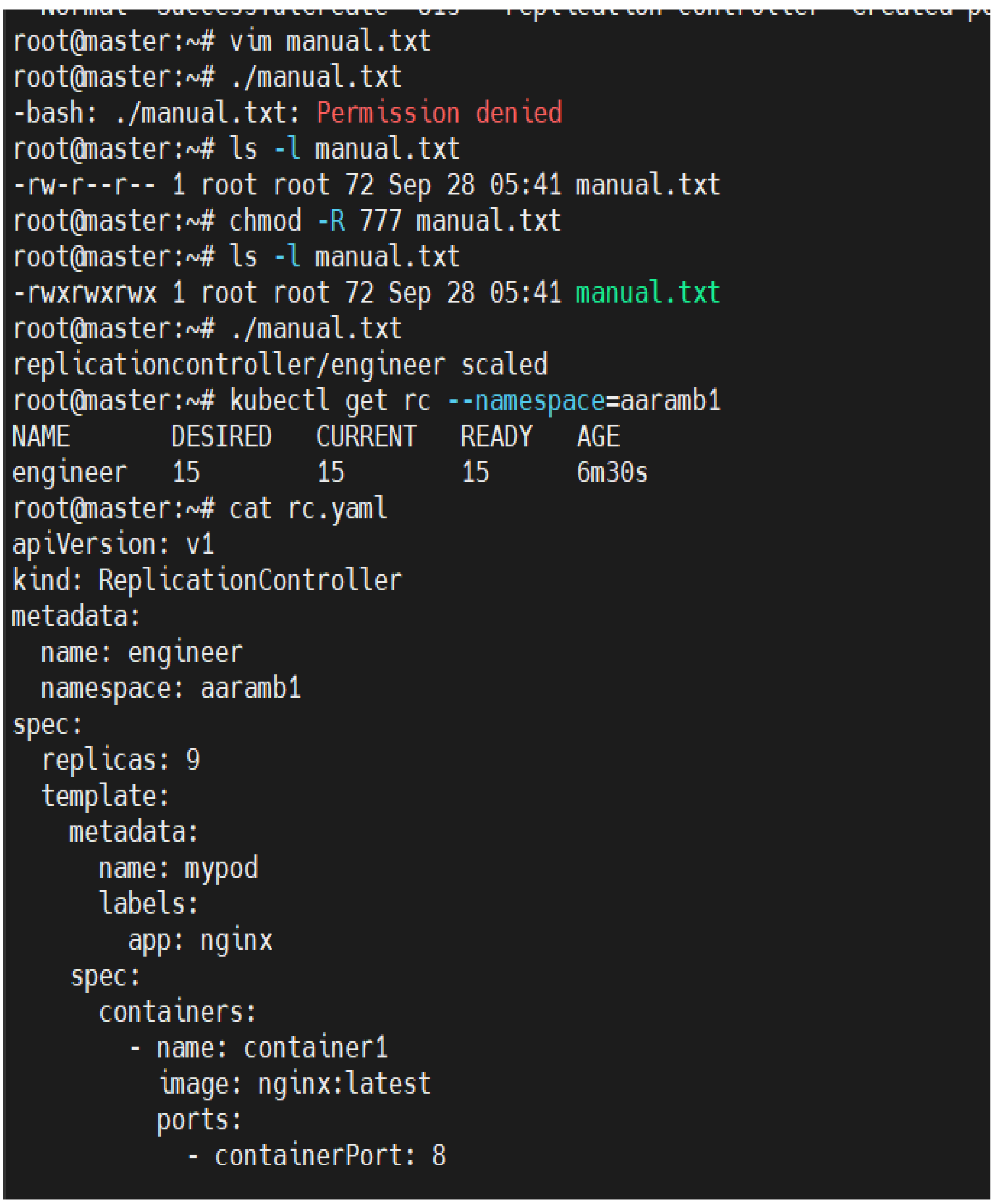
      A *Deployment* provides declarative updates
      for Pods and Replica Sets.

6. **Create an RC name engineer (it should be run in a namespace called aaramb1, rc should manage 9 replicas), create one file called manual.txt, and write a command to scale up your pod replicas to 15 without changing it in the manifest file).**



```
root@master:~# kubectl get namespace
NAME              STATUS   AGE
default           Active   4m16s
kube-node-lease   Active   4m16s
kube-public       Active   4m16s
kube-system       Active   4m16s
root@master:~# vim namespace.yaml
root@master:~# kubectl create -f namespace.yaml
namespace/aaramb1 created
root@master:~# kubectl get namespace
NAME              STATUS   AGE
aaramb1           Active   5s
default           Active   6m11s
kube-node-lease   Active   6m11s
kube-public       Active   6m11s
kube-system       Active   6m11s
root@master:~# ls
calico.yaml  namespace.yaml  snap
root@master:~# vim manual.txt
root@master:~# ls
calico.yaml  manual.txt  namespace.yaml  snap
root@master:~# vim rc.yaml
root@master:~# vim rc.yaml
root@master:~# kubectl create -f rc.yaml
replicationcontroller/engineer created
root@master:~# kubectl get rc
No resources found in default namespace.
root@master:~# kubectl get rc --namespace=aaramb1
NAME       DESIRED   CURRENT   READY   AGE
engineer   9         9         9       30s
root@master:~# kubectl descrbe rc engineer --namespace=aaramb1
error: unknown command "descrbe" for "kubectl"

Did you mean this?
        describe
root@master:~# kubectl describe rc engineer --namespace=aaramb1
Name:         engineer
Namespace:    aaramb1
Selector:     app=nginx
Labels:       app=nginx
Annotations:  <none>
Replicas:     9 current / 9 desired
```



```
  Normal  SuccessfulCreate  61s   replication-controller  Created po
root@master:~# vim manual.txt
root@master:~# ./manual.txt
-bash: ./manual.txt: Permission denied
root@master:~# ls -l manual.txt
-rw-r--r-- 1 root root 72 Sep 28 05:41 manual.txt
root@master:~# chmod -R 777 manual.txt
root@master:~# ls -l manual.txt
-rwxrwxrwx 1 root root 72 Sep 28 05:41 manual.txt
root@master:~# ./manual.txt
replicationcontroller/engineer scaled
root@master:~# kubectl get rc --namespace=aaramb1
NAME       DESIRED   CURRENT   READY   AGE
engineer   15        15        15      6m30s
root@master:~# cat rc.yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: engineer
  namespace: aaramb1
spec:
  replicas: 9
  template:
    metadata:
      name: mypod
      labels:
        app: nginx
    spec:
      containers:
        - name: container1
          image: nginx:latest
          ports:
            - containerPort: 8
```

7. What is the difference between RC and RS? What do labels and selectors explain?

Replication Controller	Replica Set
The Replication Controller uses equality-based selectors to manage the pods.	ReplicaSets Controller uses set-based selectors to manage the pods.
The rolling-update command works with Replication Controllers	The rolling-update command won't work with ReplicaSets.
Replica Controller is deprecated and replaced by ReplicaSets.	Deployments are recommended over ReplicaSets.
Selector field is not mandatory in replication controller	Selector field is mandatory in replication controller

Labels : *Labels* are key/value pairs that are attached to objects such as Pods.
Labels can be used to organize and to select subsets of objects.
Labels can be attached to objects at creation time and subsequently added and modified at any time.
Each object can have a set of key/value labels defined. Each Key must be unique for a given object.
labels do not provide uniqueness. In general, we expect many objects to carry the same label(s).
Selectors: the client/user can identify a set of objects.
The label selector is the core grouping primitive in Kubernetes.

The API currently supports two types of selectors: *equality-based* and *set-based*.

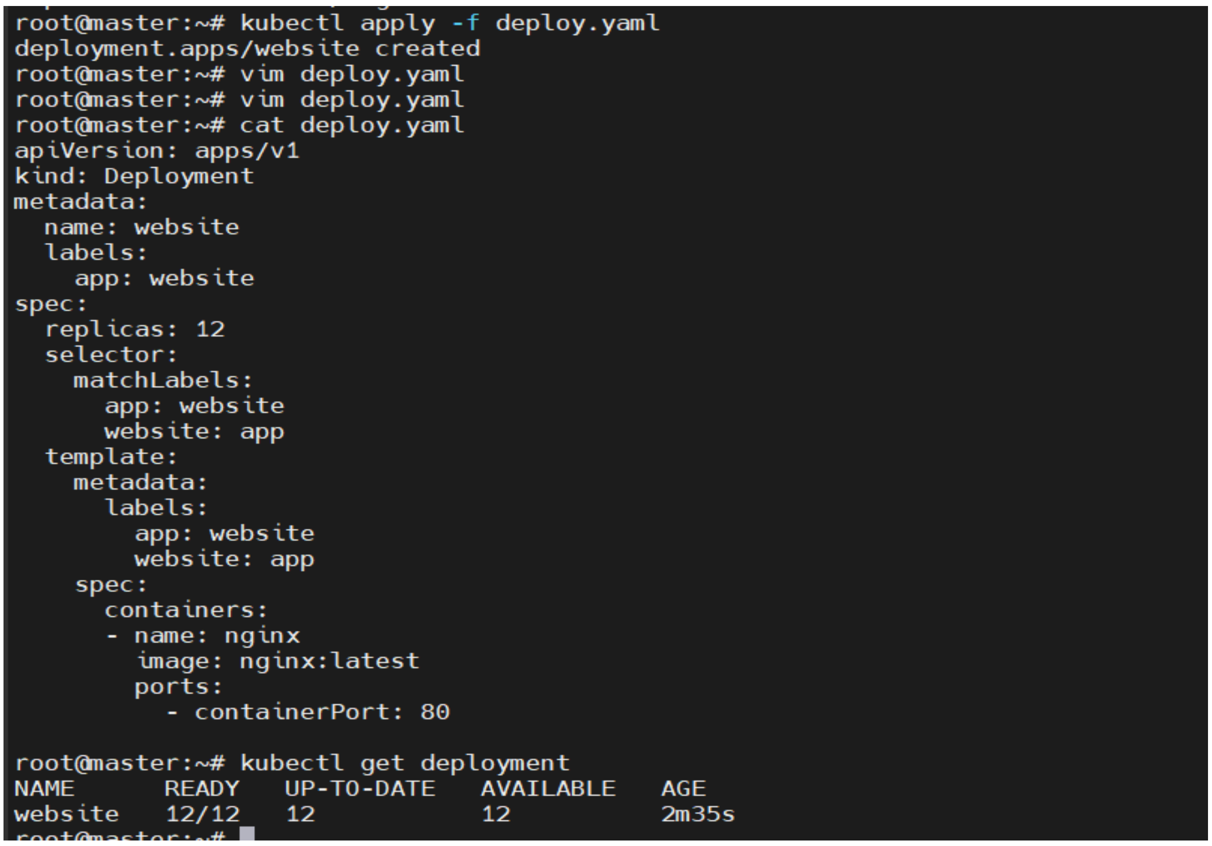
8. **Create a deployment in your system:**

   - The deployment name should be website

   - Labels it uses must be app=website and website=app

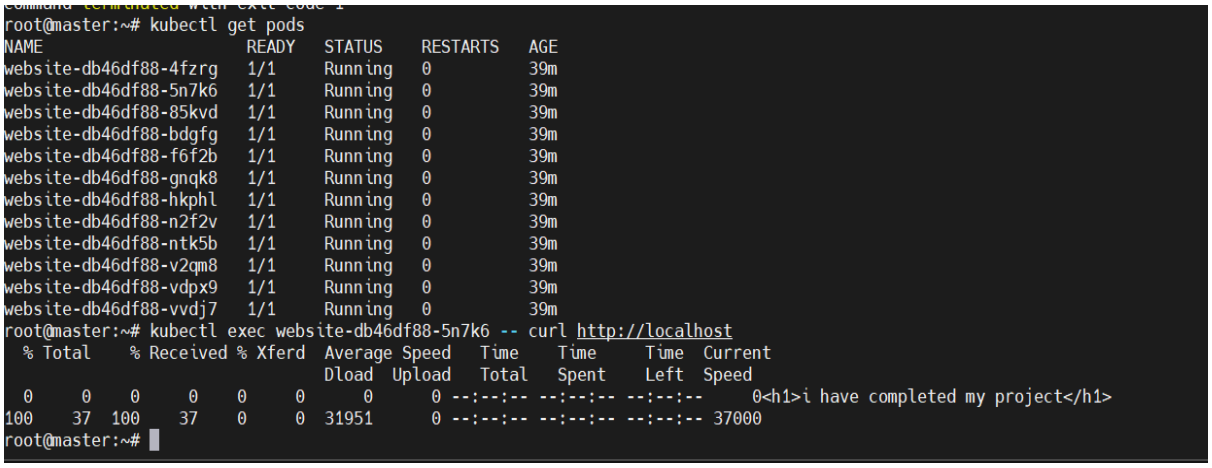
   - Replicas it should use 12

   - Use the image nginx:latest for deployment



```
root@master:~# kubectl apply -f deploy.yaml
deployment.apps/website created
root@master:~# vim deploy.yaml
root@master:~# vim deploy.yaml
root@master:~# cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website
  labels:
    app: website
spec:
  replicas: 12
  selector:
    matchLabels:
      app: website
      website: app
  template:
    metadata:
      labels:
        app: website
        website: app
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
          - containerPort: 80

root@master:~# kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
website   12/12   12           12          2m35s
```



```
command terminated with exit code 1
root@master:~# kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
website-db46df88-4fzrg    1/1     Running   0          39m
website-db46df88-5n7k6    1/1     Running   0          39m
website-db46df88-85kvd    1/1     Running   0          39m
website-db46df88-bdgfg    1/1     Running   0          39m
website-db46df88-f6f2b    1/1     Running   0          39m
website-db46df88-gnqk8    1/1     Running   0          39m
website-db46df88-hkphl    1/1     Running   0          39m
website-db46df88-n2f2v    1/1     Running   0          39m
website-db46df88-ntk5b    1/1     Running   0          39m
website-db46df88-v2qm8    1/1     Running   0          39m
website-db46df88-vdpx9    1/1     Running   0          39m
website-db46df88-vvdj7    1/1     Running   0          39m
root@master:~# kubectl exec website-db46df88-5n7k6 -- curl http://localhost
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<h1>i have completed my project</h1>
100    37  100    37    0     0  31951      0 --:--:-- --:--:-- --:--:-- 37000
root@master:~# 
```