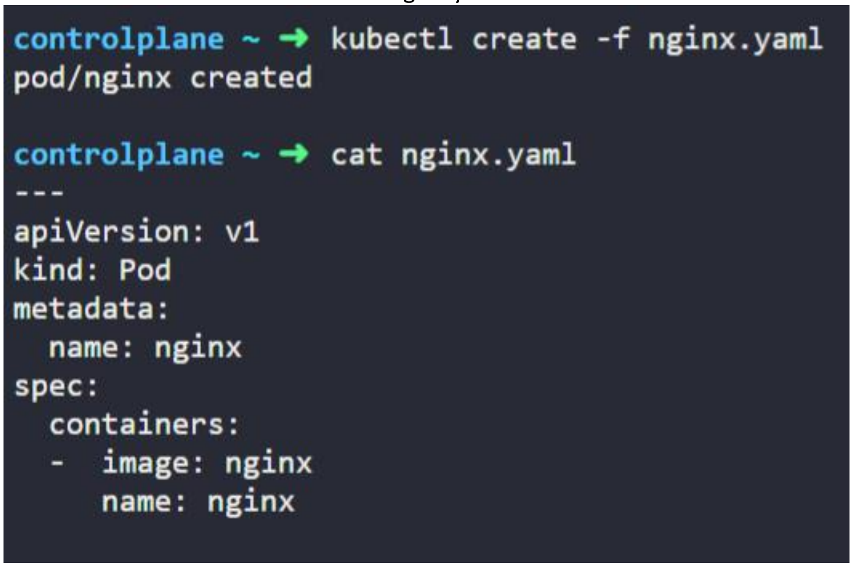
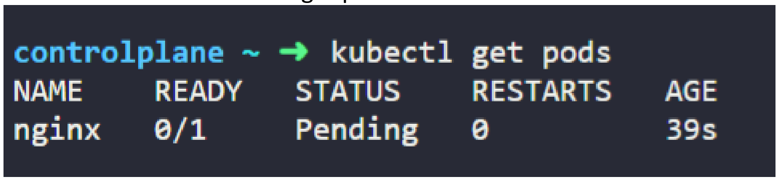
A pod definition file nginx.yaml is given. Create a pod using the file.
Only create the POD for now. We will inspect its status next.
Use the command kubectl create -f nginx.yaml



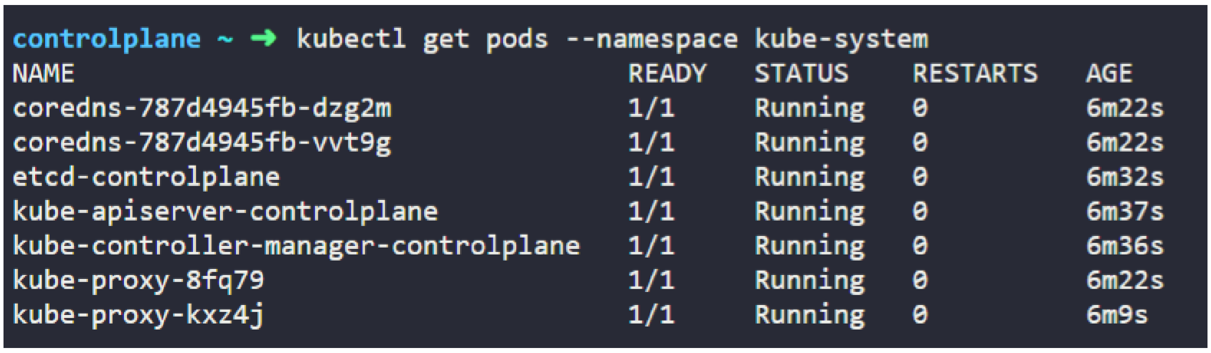
```
controlplane ~ ➜ kubectl create -f nginx.yaml
pod/nginx created

controlplane ~ ➜ cat nginx.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  -  image: nginx
     name: nginx
```

What is the status of the created POD?
Run the command: kubectl get pods and check the Status column.



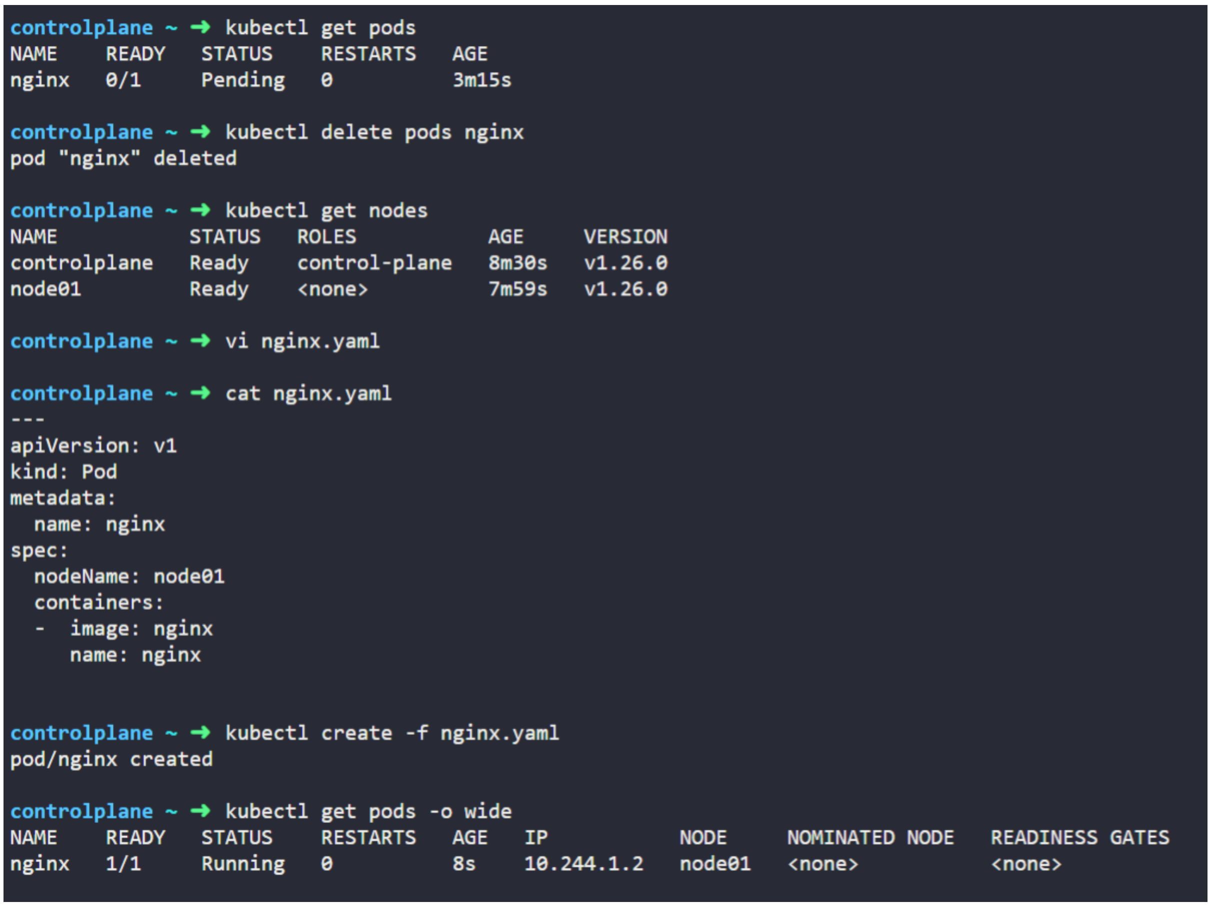
```
controlplane ~ ➜ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          39s
```

Why is the POD in a pending state? No scheduler present
Inspect the environment for various kubernetes control plane components.
Run the command: kubectl get pods --namespace kube-system to see the status of scheduler pod. We have removed the scheduler from this Kubernetes cluster. As a result, as it stands, the pod will remain in a pending state forever.



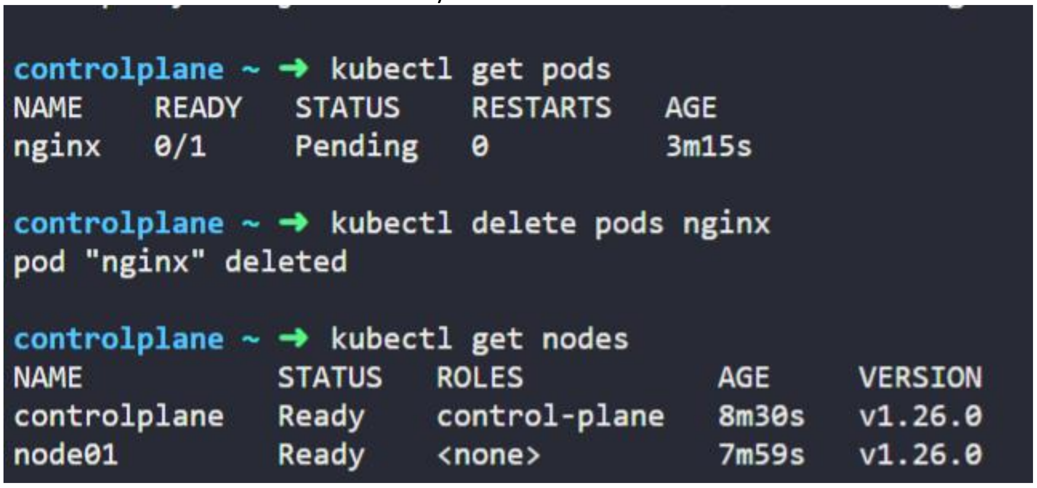
```
controlplane ~ ➜ kubectl get pods --namespace kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-787d4945fb-dzg2m               1/1     Running   0          6m22s
coredns-787d4945fb-vvt9g               1/1     Running   0          6m22s
etcd-controlplane                      1/1     Running   0          6m32s
kube-apiserver-controlplane            1/1     Running   0          6m37s
kube-controller-manager-controlplane   1/1     Running   0          6m36s
kube-proxy-8fq79                       1/1     Running   0          6m22s
kube-proxy-kxz4j                       1/1     Running   0          6m9s
```

Manually schedule the pod on node01.
Delete and recreate the POD if necessary.

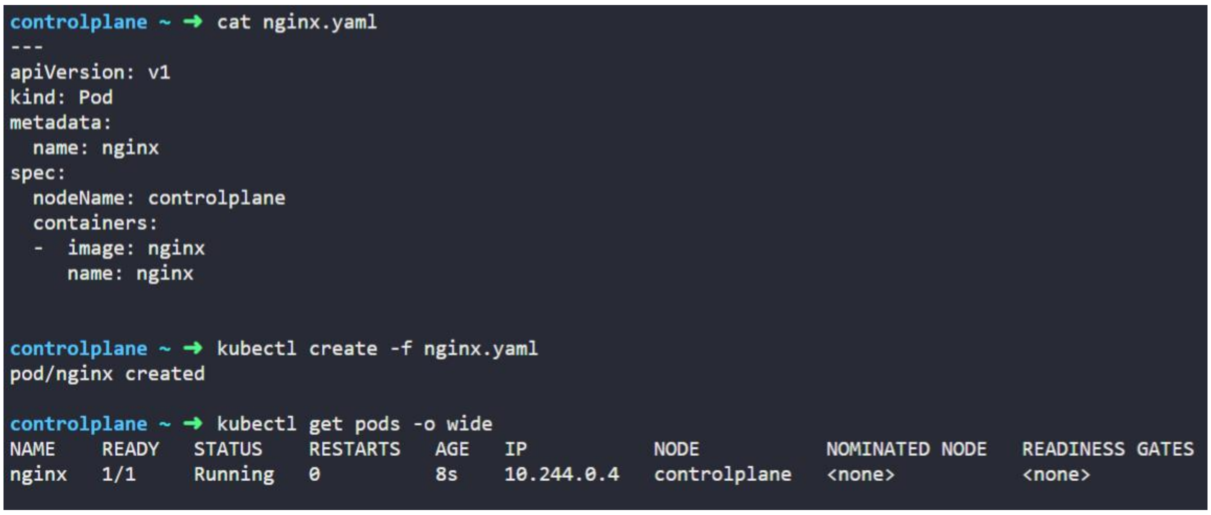


```
controlplane ~ ➜ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          3m15s

controlplane ~ ➜ kubectl delete pods nginx
pod "nginx" deleted

controlplane ~ ➜ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
controlplane   Ready    control-plane   8m30s   v1.26.0
node01         Ready    <none>          7m59s   v1.26.0

controlplane ~ ➜ vi nginx.yaml

controlplane ~ ➜ cat nginx.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: node01
  containers:
  -  image: nginx
     name: nginx


controlplane ~ ➜ kubectl create -f nginx.yaml
pod/nginx created

controlplane ~ ➜ kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          8s    10.244.1.2   node01   <none>           <none>
```

Now schedule the same pod on the controlplane node.
Delete and recreate the POD if necessary.

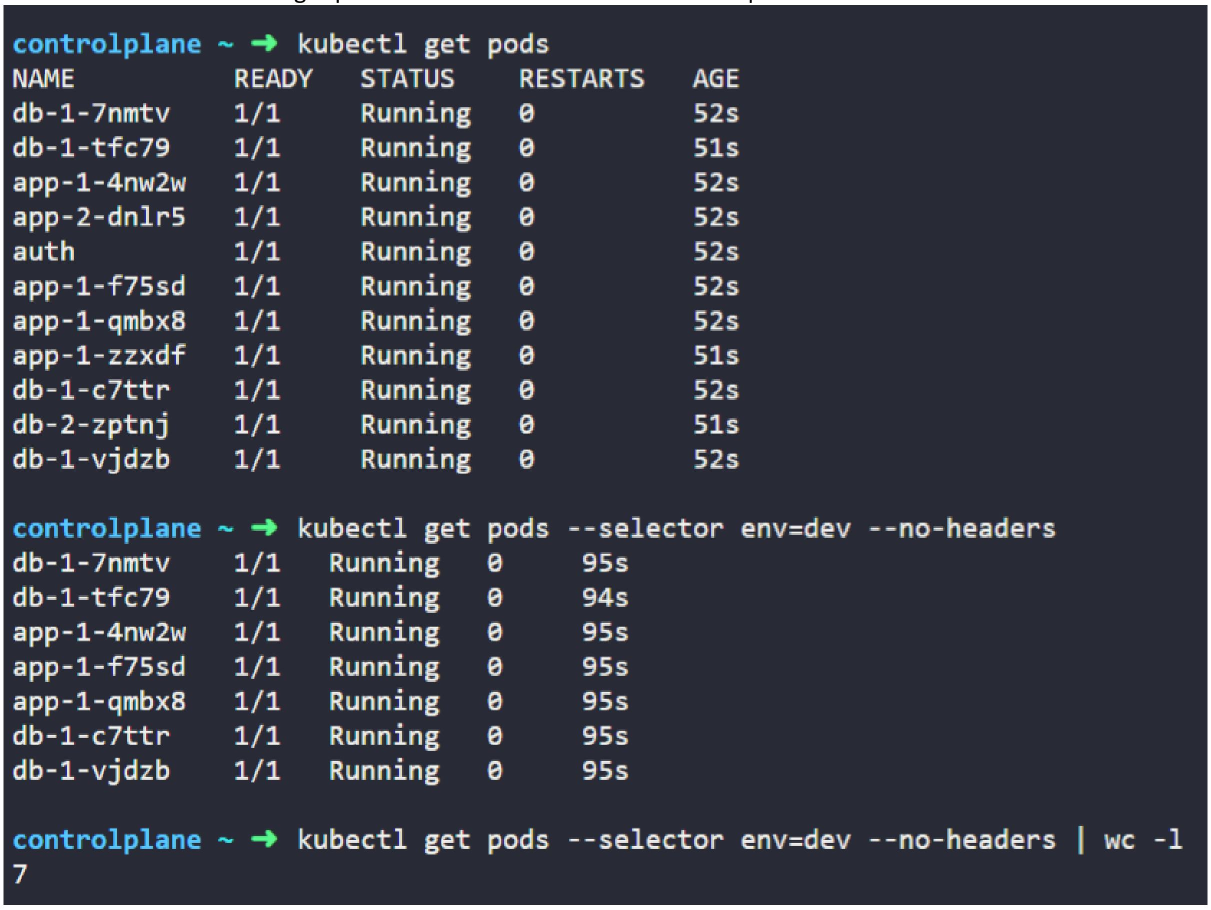


```
controlplane ~ ➜ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          3m15s

controlplane ~ ➜ kubectl delete pods nginx
pod "nginx" deleted

controlplane ~ ➜ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
controlplane   Ready    control-plane   8m30s   v1.26.0
node01         Ready    <none>          7m59s   v1.26.0
```



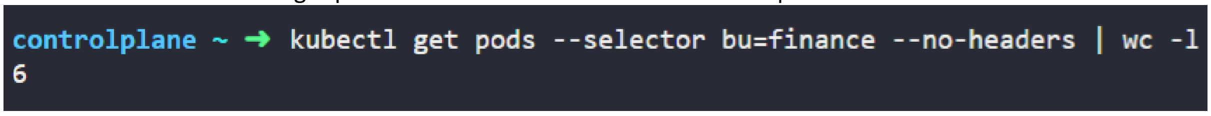
```
controlplane ~ ➜ cat nginx.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: controlplane
  containers:
  -  image: nginx
     name: nginx


controlplane ~ ➜ kubectl create -f nginx.yaml
pod/nginx created

controlplane ~ ➜ kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE           NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          8s    10.244.0.4   controlplane   <none>           <none>
```

Labels and selectors :
We have deployed a number of PODs. They are labelled with tier, env and bu. How many PODs exist in the dev environment (env)?
Use selectors to filter the output
Run the command kubectl get pods --selector env=dev --no-headers | wc -l



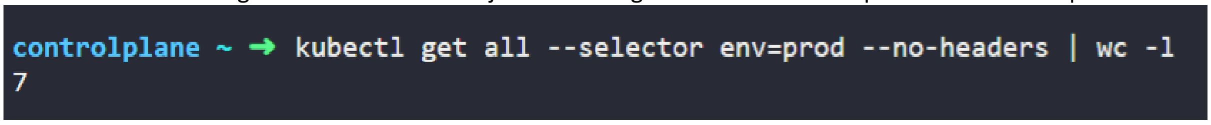
```
controlplane ~ ➜ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-1-7nmtv    1/1     Running   0          52s
db-1-tfc79    1/1     Running   0          51s
app-1-4nw2w   1/1     Running   0          52s
app-2-dnlr5   1/1     Running   0          52s
auth          1/1     Running   0          52s
app-1-f75sd   1/1     Running   0          52s
app-1-qmbx8   1/1     Running   0          52s
app-1-zzxdf   1/1     Running   0          51s
db-1-c7ttr    1/1     Running   0          52s
db-2-zptnj    1/1     Running   0          51s
db-1-vjdzb    1/1     Running   0          52s

controlplane ~ ➜ kubectl get pods --selector env=dev --no-headers
db-1-7nmtv    1/1   Running   0     95s
db-1-tfc79    1/1   Running   0     94s
app-1-4nw2w   1/1   Running   0     95s
app-1-f75sd   1/1   Running   0     95s
app-1-qmbx8   1/1   Running   0     95s
db-1-c7ttr    1/1   Running   0     95s
db-1-vjdzb    1/1   Running   0     95s

controlplane ~ ➜ kubectl get pods --selector env=dev --no-headers | wc -l
7
```

How many PODs are in the finance business unit (bu)?
Run the command kubectl get pods --selector bu=finance --no-headers | wc -l



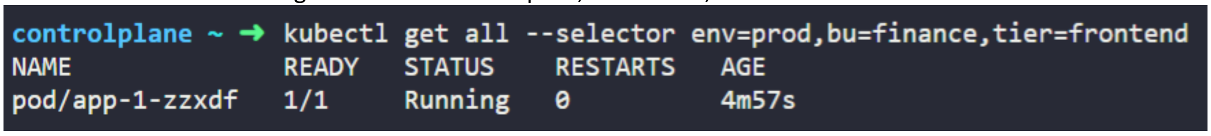
```
controlplane ~ ➜ kubectl get pods --selector bu=finance --no-headers | wc -l
6
```

How many objects are in the prod environment including PODs, ReplicaSets and any other objects?
Run the command to get exact number of objects kubectl get all --selector env=prod --no-headers | wc -l



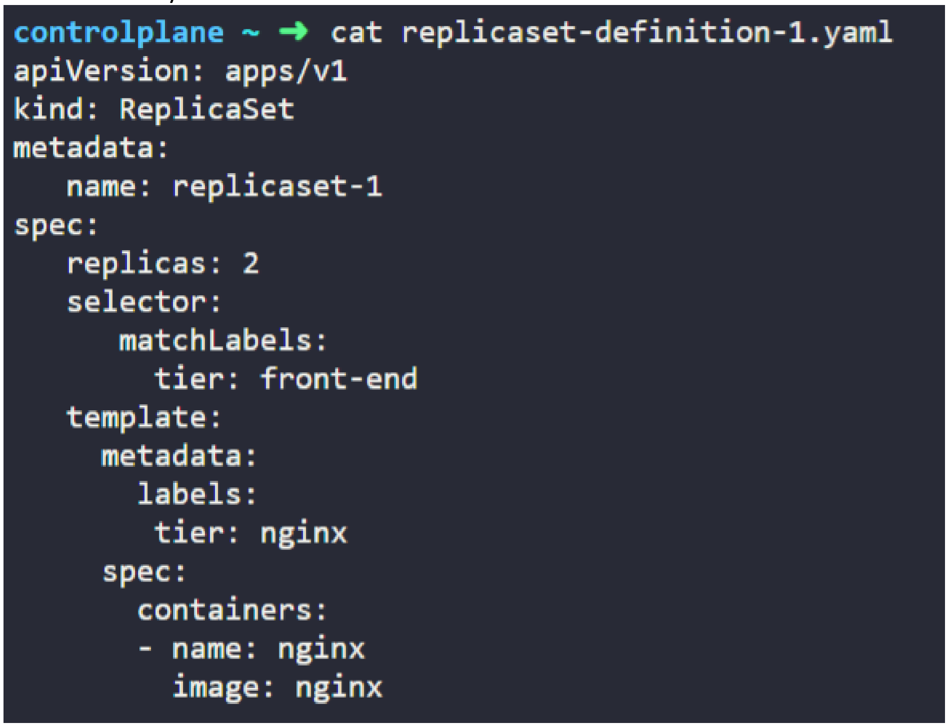
```
controlplane ~ ➜ kubectl get all --selector env=prod --no-headers | wc -l
7
```

Identify the POD which is part of the prod environment, the finance BU and of frontend tier?
Run the command kubectl get all --selector env=prod,bu=finance,tier=frontend

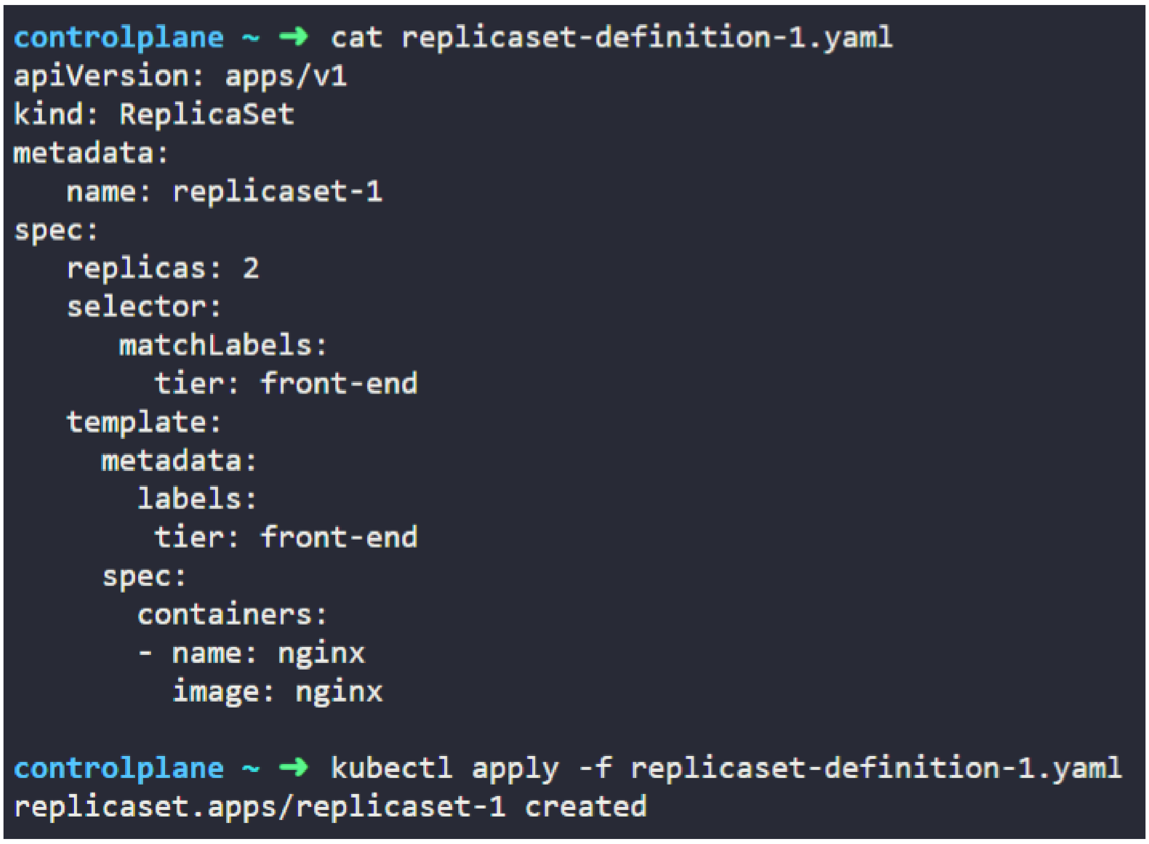


```
controlplane ~ ➜ kubectl get all --selector env=prod,bu=finance,tier=frontend
NAME               READY   STATUS    RESTARTS   AGE
pod/app-1-zzxdf    1/1     Running   0          4m57s
```

A ReplicaSet definition file is given replicaset-definition-1.yaml. Try to create the replicaset. There is an issue with the file. Try to fix it.

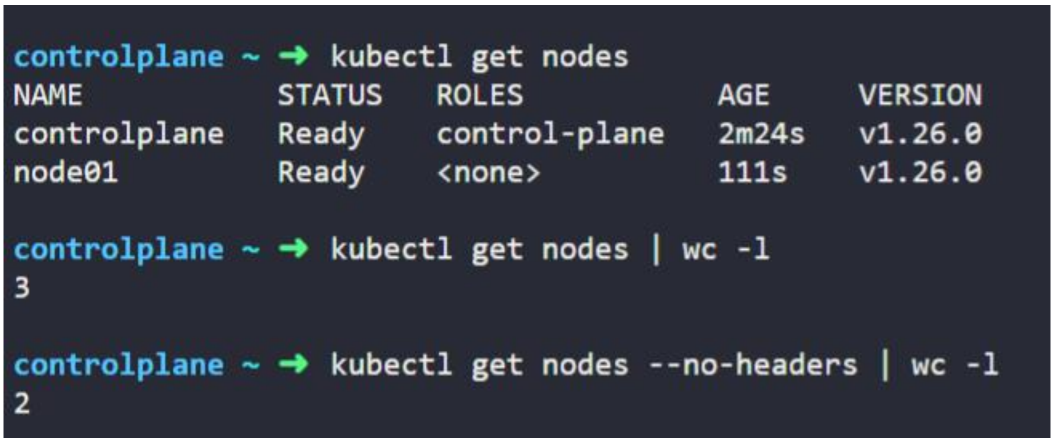


```
controlplane ~ ➜ cat replicaset-definition-1.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
   name: replicaset-1
spec:
   replicas: 2
   selector:
      matchLabels:
        tier: front-end
   template:
     metadata:
       labels:
        tier: nginx
     spec:
       containers:
       - name: nginx
         image: nginx
```



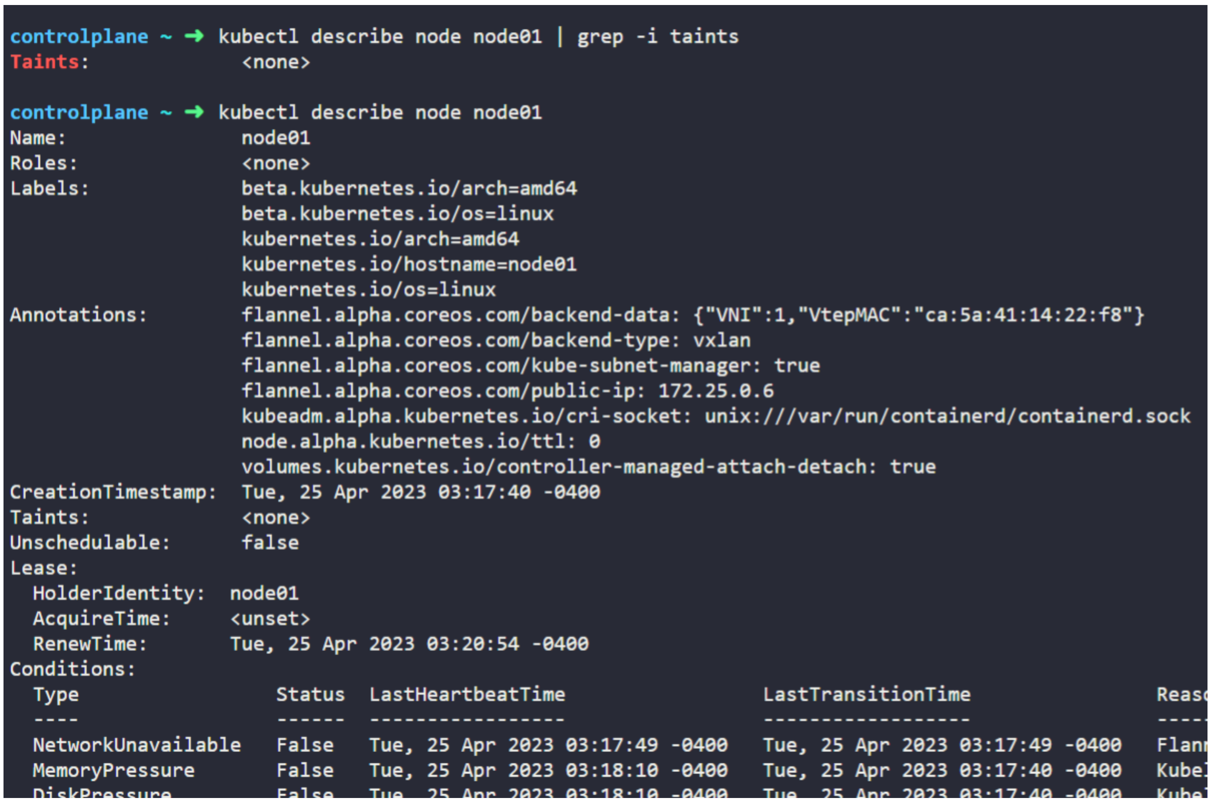
```
controlplane ~ ➜  cat replicaset-definition-1.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
   name: replicaset-1
spec:
   replicas: 2
   selector:
      matchLabels:
        tier: front-end
   template:
     metadata:
       labels:
        tier: front-end
     spec:
       containers:
       - name: nginx
         image: nginx

controlplane ~ ➜  kubectl apply -f replicaset-definition-1.yaml
replicaset.apps/replicaset-1 created
```

Taints and Tolerations



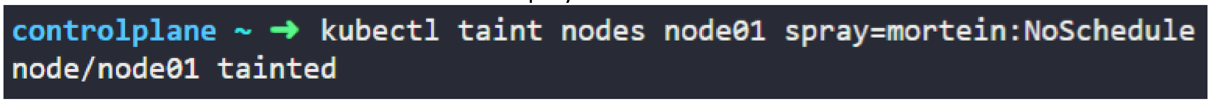
```
controlplane ~ ➜  kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
controlplane   Ready    control-plane   2m24s   v1.26.0
node01         Ready    <none>          111s    v1.26.0

controlplane ~ ➜  kubectl get nodes | wc -l
3

controlplane ~ ➜  kubectl get nodes --no-headers | wc -l
2
```

Do any taints exist on node01 node?
Run the command: kubectl describe node node01 | grep -i taints to check taint exists



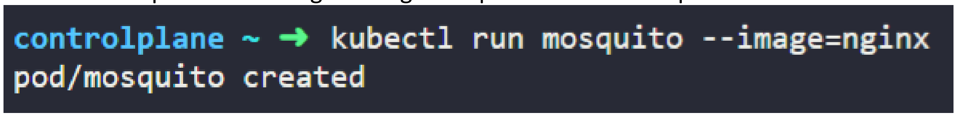
```
controlplane ~ ➜ kubectl describe node node01 | grep -i taints
Taints:             <none>

controlplane ~ ➜ kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"ca:5a:41:14:22:f8"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.25.0.6
                    kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 25 Apr 2023 03:17:40 -0400
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  node01
  AcquireTime:     <unset>
  RenewTime:       Tue, 25 Apr 2023 03:20:54 -0400
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reas
  ----                 ------  -----------------                 ------------------                ----
  NetworkUnavailable   False   Tue, 25 Apr 2023 03:17:49 -0400   Tue, 25 Apr 2023 03:17:49 -0400   Flan
  MemoryPressure       False   Tue, 25 Apr 2023 03:18:10 -0400   Tue, 25 Apr 2023 03:17:40 -0400   Kube
  DiskPressure         False   Tue, 25 Apr 2023 03:18:10 -0400   Tue, 25 Apr 2023 03:17:40 -0400   Kube
```

Create a taint on node01 with key of spray, value of mortein and effect of NoSchedule
Run the command: kubectl taint nodes node01 spray=mortein:NoSchedule



```
controlplane ~ ➜ kubectl taint nodes node01 spray=mortein:NoSchedule
node/node01 tainted
```

Create a new pod with the nginx image and pod name as mosquito.



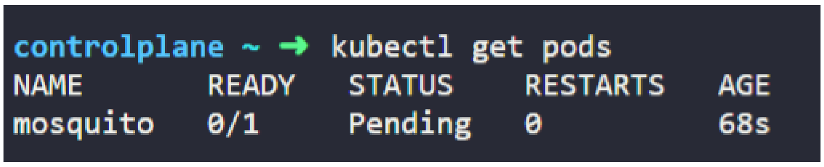
```
controlplane ~ ➜ kubectl run mosquito --image=nginx
pod/mosquito created
```

Solution manifest file to create a pod called mosquito as follows:

```
---
apiVersion: v1
kind: Pod
metadata:
  name: mosquito
spec:
  containers:
  - image: nginx
    name: mosquito
```

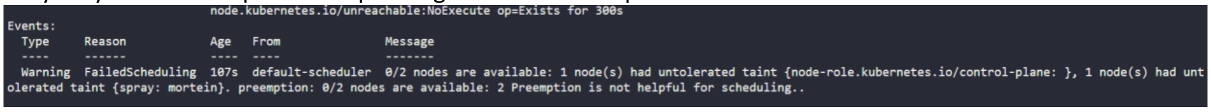
then run kubectl create -f <FILE-NAME>.yaml

What is the state of the POD?



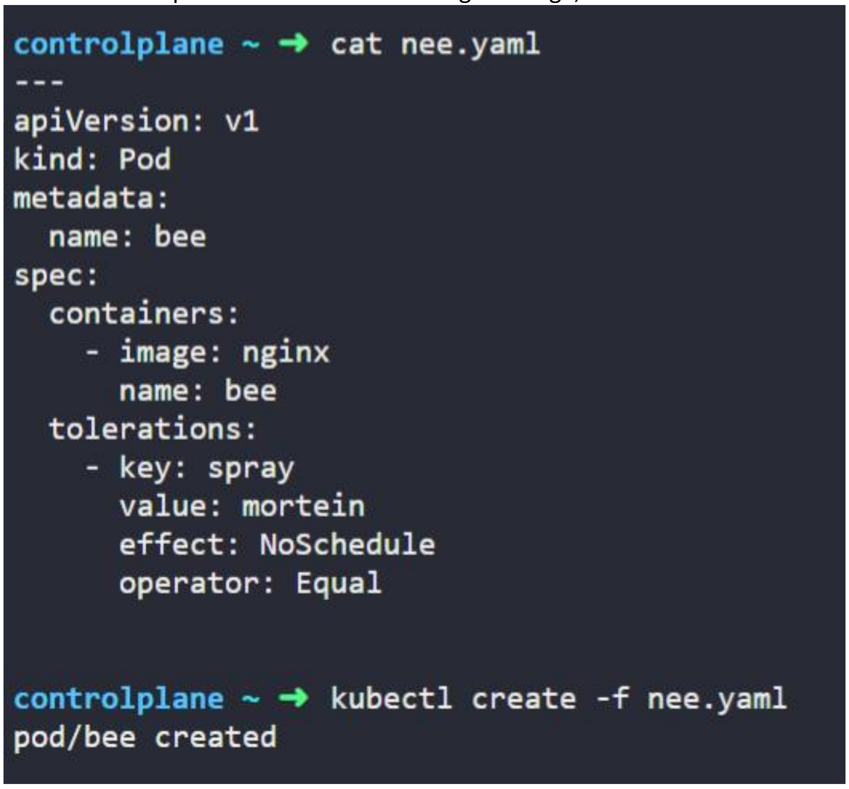
```
controlplane ~ ➜ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          68s
```

Why do you think the pod is in a pending state? Pod mosqioto cont tolerate taint mortein .



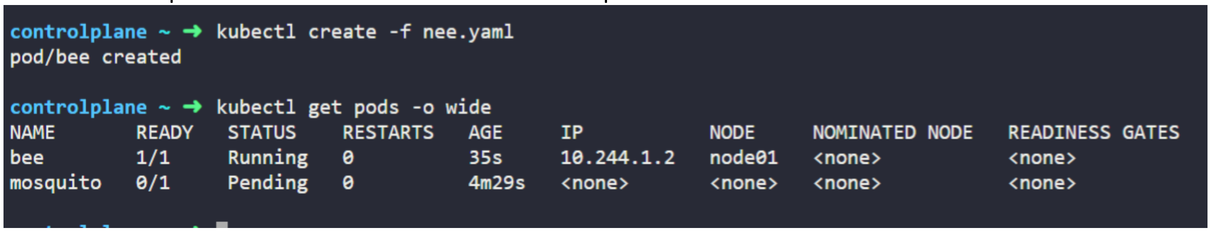
```
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  107s  default-scheduler  0/2 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/control-plane: }, 1 node(s) had unt
olerated taint {spray: mortein}. preemption: 0/2 nodes are available: 2 Preemption is not helpful for scheduling..
```

Create another pod named bee with the nginx image, which has a toleration set to the taint mortein.



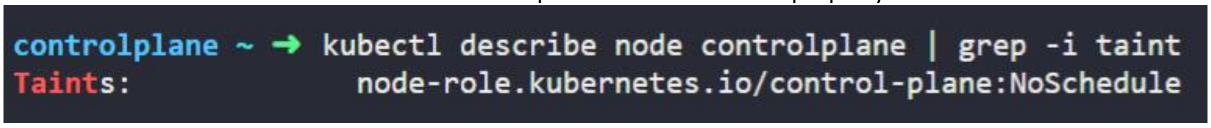
```
controlplane ~ ➜  cat nee.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: bee
spec:
  containers:
    - image: nginx
      name: bee
  tolerations:
    - key: spray
      value: mortein
      effect: NoSchedule
      operator: Equal


controlplane ~ ➜  kubectl create -f nee.yaml
pod/bee created
```

Notice the bee pod was scheduled on node node01 despite the taint.



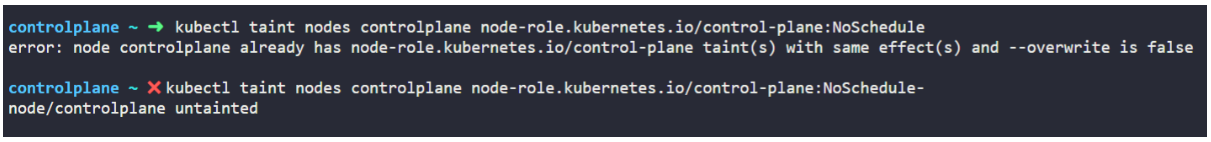
```
controlplane ~ ➜  kubectl create -f nee.yaml
pod/bee created

controlplane ~ ➜  kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP           NODE     NOMINATED NODE   READINESS GATES
bee        1/1     Running   0          35s     10.244.1.2   node01   <none>           <none>
mosquito   0/1     Pending   0          4m29s   <none>       <none>   <none>           <none>
```

Do you see any taints on controlplane node?
Run the command: kubectl describe node controlplane and see the taint property.



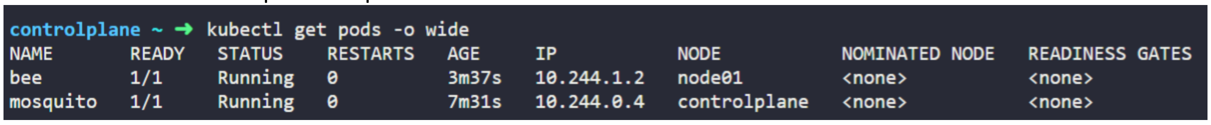
```
controlplane ~ ➜  kubectl describe node controlplane | grep -i taint
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
```

Remove the taint on controlplane, which currently has the taint effect of NoSchedule.
Run the command: kubectl taint nodes controlplane node-role.kubernetes.io/control-plane:NoSchedule- to untaint the node.



```
controlplane ~ ➜  kubectl taint nodes controlplane node-role.kubernetes.io/control-plane:NoSchedule
error: node controlplane already has node-role.kubernetes.io/control-plane taint(s) with same effect(s) and --overwrite is false

controlplane ~ ✗ kubectl taint nodes controlplane node-role.kubernetes.io/control-plane:NoSchedule-
node/controlplane untainted
```

What is the state of the pod mosquito now?

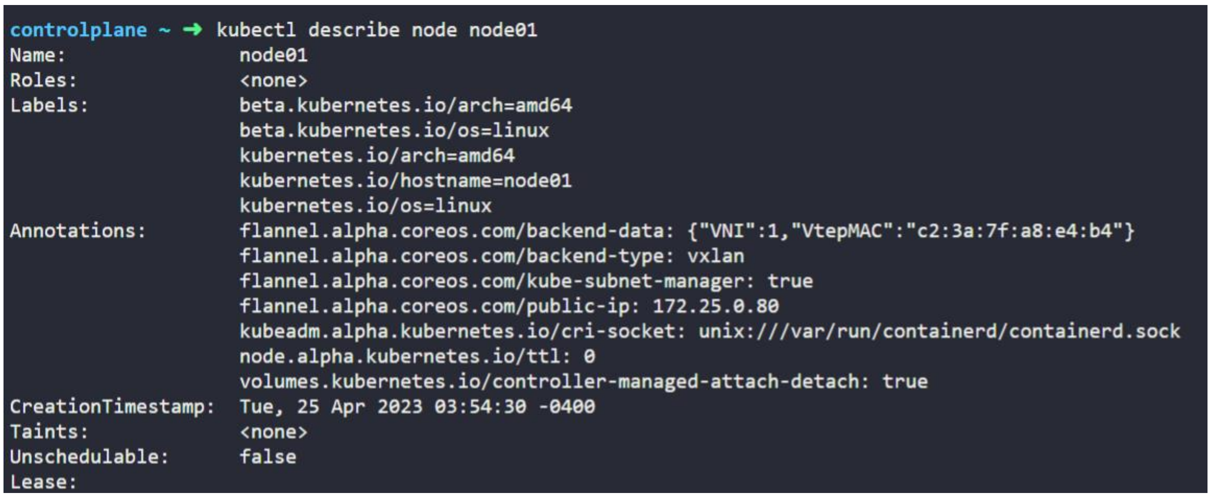


```
controlplane ~ ➜  kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATED NODE   READINESS GATES
bee        1/1     Running   0          3m37s   10.244.1.2   node01         <none>           <none>
mosquito   1/1     Running   0          7m31s   10.244.0.4   controlplane   <none>           <none>
```

Which node is the POD mosquito on now?
Controlplane

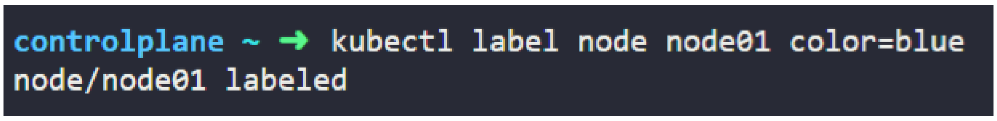
NodeAffinity

How many Labels exist on node node01?
Run the command kubectl describe node node01 and count the number of labels.



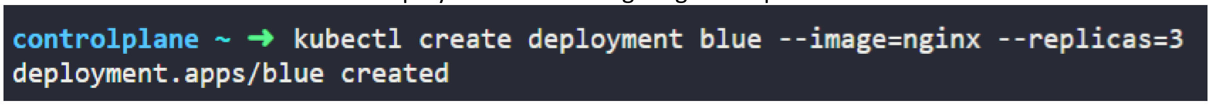
```
controlplane ~ ➜  kubectl describe node node01
Name:               node01
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node01
                    kubernetes.io/os=linux
Annotations:        flannel.alpha.coreos.com/backend-data: {"VNI":1,"VtepMAC":"c2:3a:7f:a8:e4:b4"}
                    flannel.alpha.coreos.com/backend-type: vxlan
                    flannel.alpha.coreos.com/kube-subnet-manager: true
                    flannel.alpha.coreos.com/public-ip: 172.25.0.80
                    kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Tue, 25 Apr 2023 03:54:30 -0400
Taints:             <none>
Unschedulable:      false
Lease:
```

What is the value set to the label key beta.kubernetes.io/arch on node01?
Apply a label color=blue to node node01
Run the command: kubectl label node node01 color=blue



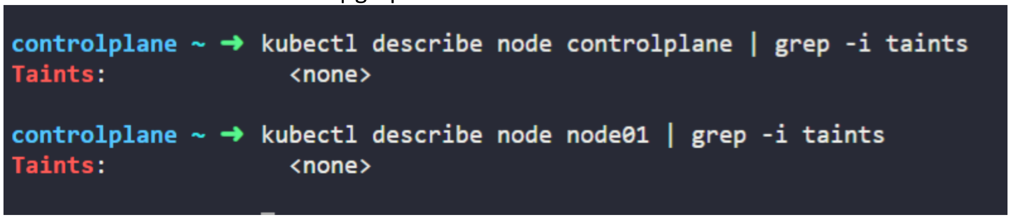
```
controlplane ~ ➜  kubectl label node node01 color=blue
node/node01 labeled
```

Create a new deployment named blue with the nginx image and 3 replicas.
Run the command: kubectl create deployment blue --image=nginx --replicas=3



```
controlplane ~ ➜  kubectl create deployment blue --image=nginx --replicas=3
deployment.apps/blue created
```

Which nodes can the pods for the blue deployment be placed on? Controlplane and node01
Make sure to check taints on both nodes!
Check if controlplane and node01 have any taints on them that will prevent the pods to be scheduled on them.
If there are no taints, the pods can be scheduled on either node.
So run the following command to check the taints on both nodes.
kubectl describe node controlplane | grep -i taints
kubectl describe node node01 | grep -i taints

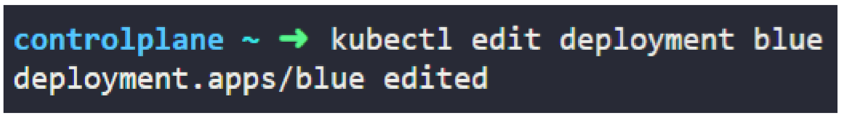


```
controlplane ~ ➜  kubectl describe node controlplane | grep -i taints
Taints:             <none>

controlplane ~ ➜  kubectl describe node node01 | grep -i taints
Taints:             <none>
```

Set Node Affinity to the deployment to place the pods on node01 only.
Edit the deployment blue and add the Node Affinity with specified key and value.

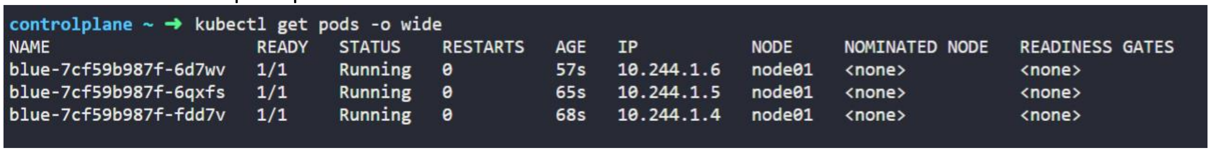
Update the deployment by running kubectl edit deployment blue and add the nodeaffinity section as follows:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blue
spec:
  replicas: 3
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: color
                operator: In
                values:
                - blue
```



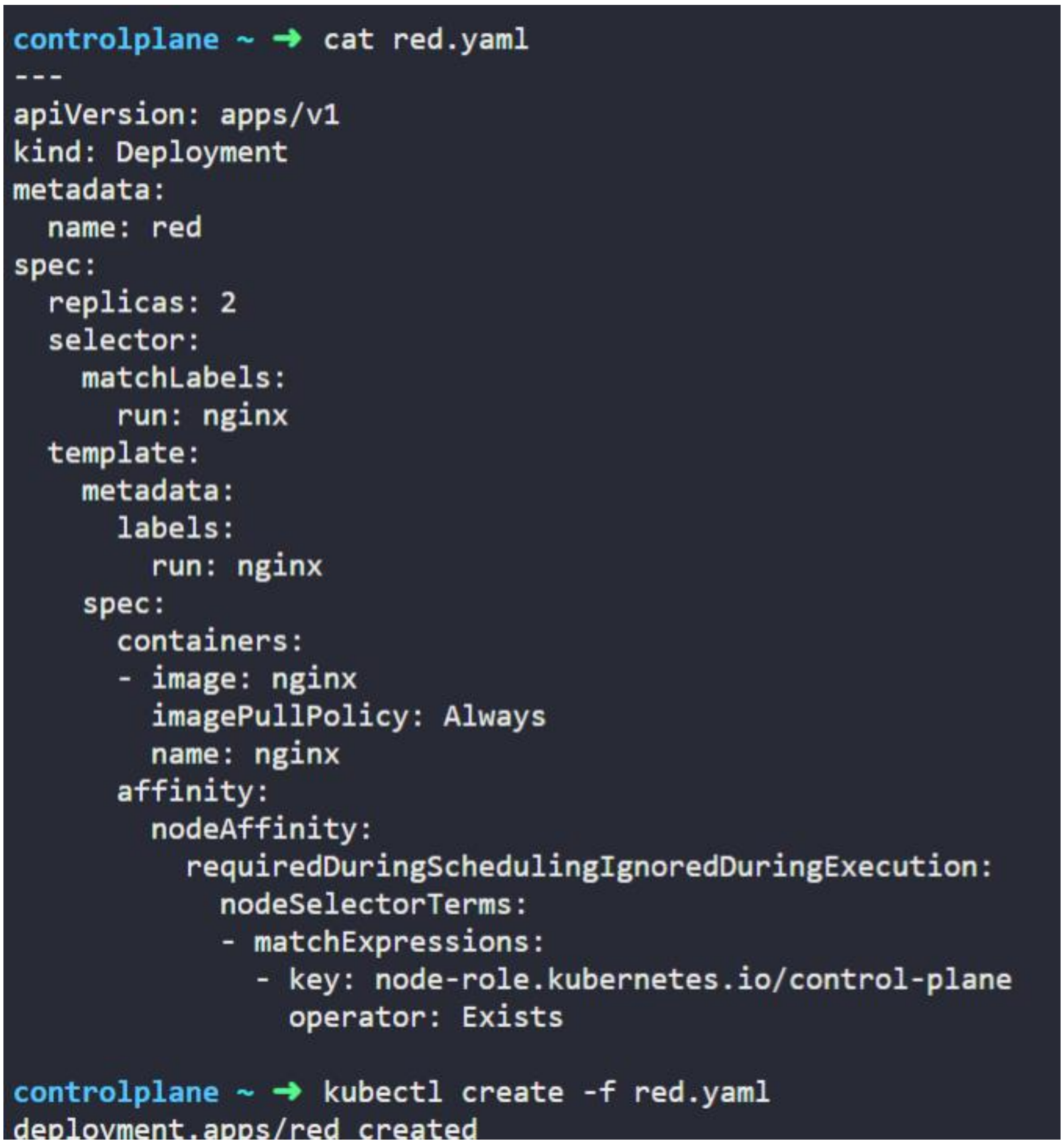
```
controlplane ~ ➜  kubectl edit deployment blue
deployment.apps/blue edited
```

Which nodes are the pods placed on now?



```
controlplane ~ ➜  kubectl get pods -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
blue-7cf59b987f-6d7wv  1/1     Running   0          57s   10.244.1.6   node01   <none>           <none>
blue-7cf59b987f-6qxfs  1/1     Running   0          65s   10.244.1.5   node01   <none>           <none>
blue-7cf59b987f-fdd7v  1/1     Running   0          68s   10.244.1.4   node01   <none>           <none>
```

Create a new deployment named red with the nginx image and 2 replicas, and ensure it gets placed on the controlplane node only.
Use the label key - node-role.kubernetes.io/control-plane - which is already set on the controlplane node.



```
controlplane ~ ➜ cat red.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: red
spec:
  replicas: 2
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: node-role.kubernetes.io/control-plane
                operator: Exists

controlplane ~ ➜ kubectl create -f red.yaml
deployment.apps/red created
```