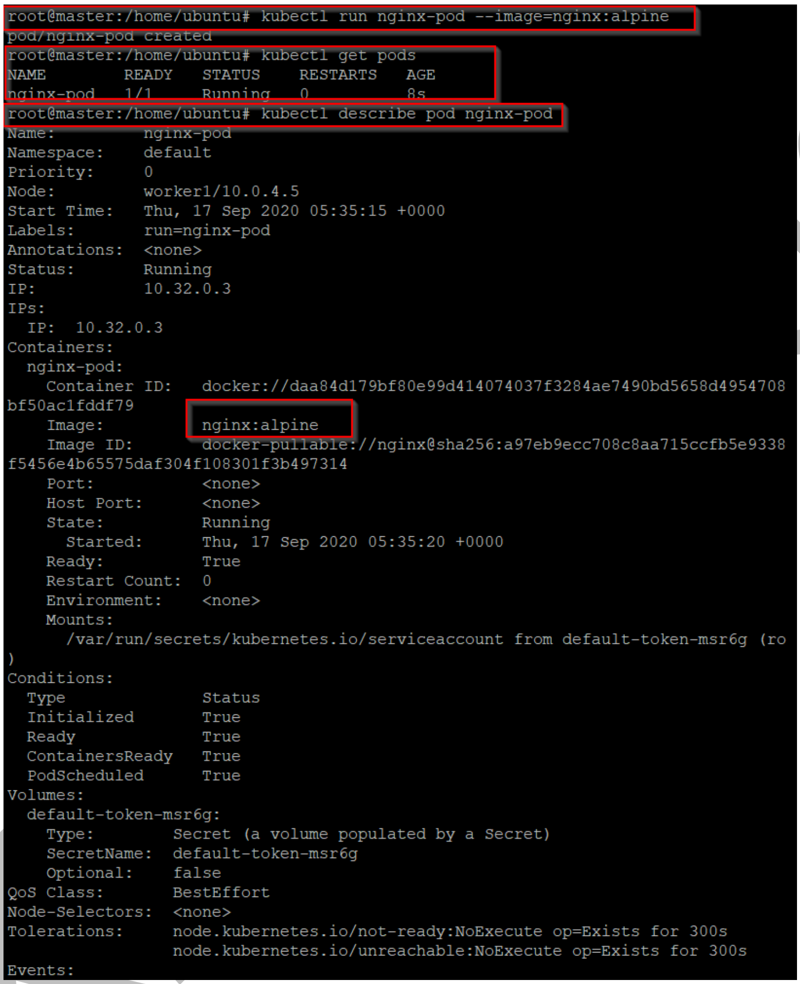
1 CKAD PRACTICE QUESTION

Note: Use Kubernetes Official Documentation to Create Object Files. https://kubernetes.io/

Q1) Deploy a pod named **nginx-pod** using the **nginx:alpine** image.

Ans:

```
kubectl run nginx-pod --image=nginx:alpine
```

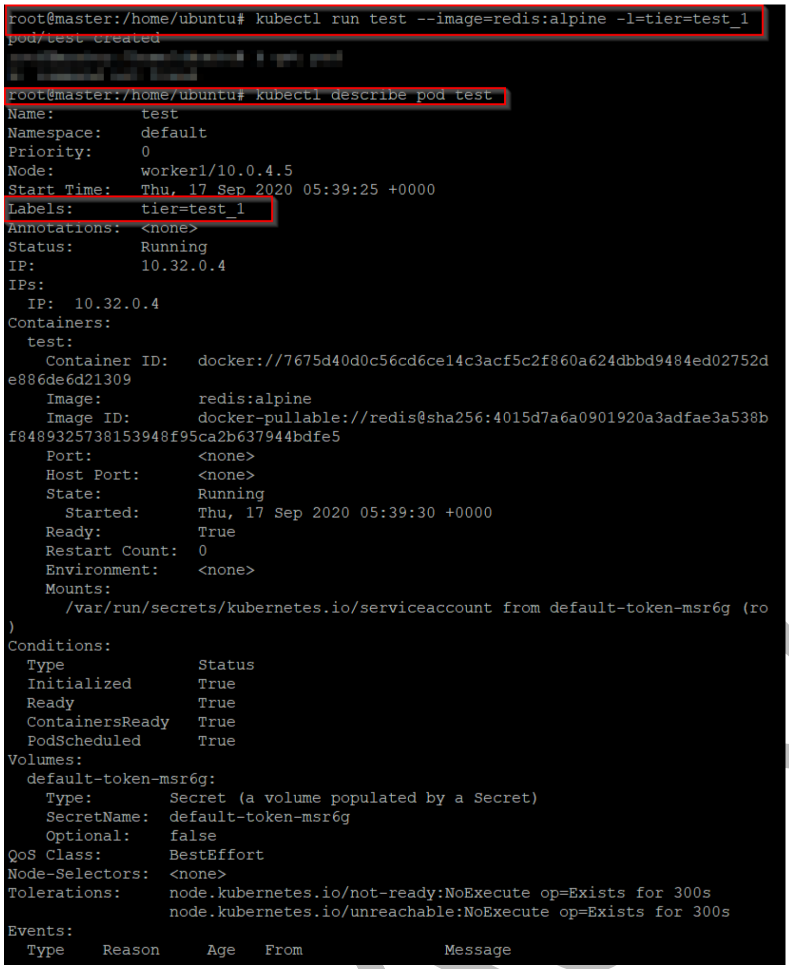


```
root@master:/home/ubuntu# kubectl run nginx-pod --image=nginx:alpine
pod/nginx-pod created
root@master:/home/ubuntu# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          8s
root@master:/home/ubuntu# kubectl describe pod nginx-pod
Name:         nginx-pod
Namespace:    default
Priority:     0
Node:         worker1/10.0.4.5
Start Time:   Thu, 17 Sep 2020 05:35:15 +0000
Labels:       run=nginx-pod
Annotations:  <none>
Status:       Running
IP:           10.32.0.3
IPs:
  IP:  10.32.0.3
Containers:
  nginx-pod:
    Container ID:   docker://daa84d179bf80e99d414074037f3284ae7490bd5658d4954708
bf50ac1fddf79
    Image:          nginx:alpine
    Image ID:       docker-pullable://nginx@sha256:a97eb9ecc708c8aa715ccfb5e9338
f5456e4b65575daf304f108301f3b497314
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 17 Sep 2020 05:35:20 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-msr6g (ro
)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-msr6g:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-msr6g
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
```

Q2) Deploy a **test** pod using the **redis:alpine** image with the labels set to **tier=test_1**.

Ans:

```
kubectl run test --image=redis:alpine -l=tier=test_1
```

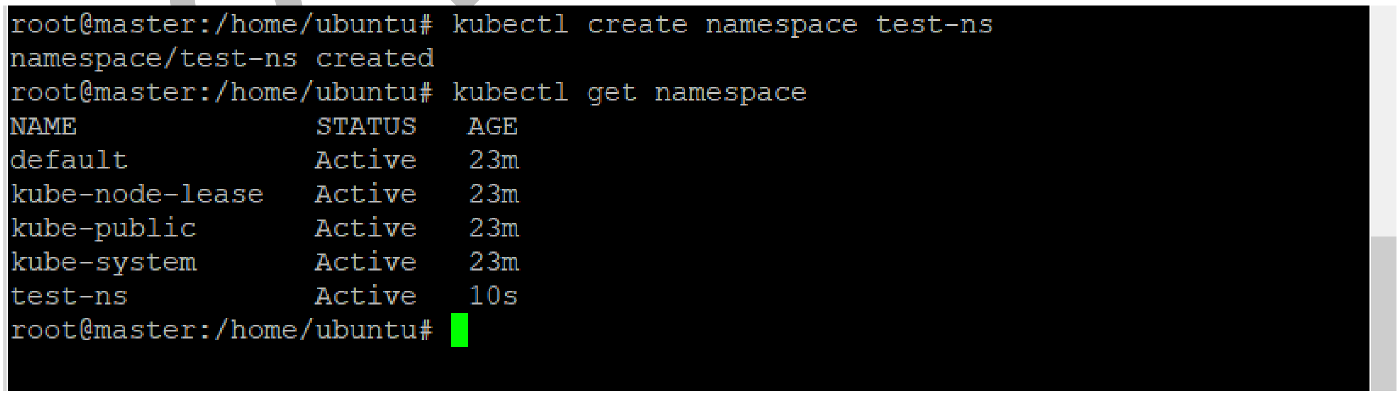


```
root@master:/home/ubuntu# kubectl run test --image=redis:alpine -l=tier=test_1
pod/test created
[redacted]
root@master:/home/ubuntu# kubectl describe pod test
Name:         test
Namespace:    default
Priority:     0
Node:         worker1/10.0.4.5
Start Time:   Thu, 17 Sep 2020 05:39:25 +0000
Labels:       tier=test_1
Annotations:  <none>
Status:       Running
IP:           10.32.0.4
IPs:
  IP:  10.32.0.4
Containers:
  test:
    Container ID:   docker://7675d40d0c56cd6ce14c3acf5c2f860a624dbbd9484ed02752d
e886de6d21309
    Image:          redis:alpine
    Image ID:       docker-pullable://redis@sha256:4015d7a6a0901920a3adfae3a538b
f8489325738153948f95ca2b637944bdfe5
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 17 Sep 2020 05:39:30 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-msr6g (ro
)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-msr6g:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-msr6g
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
```

Q3) Create a namespace named **test_ns**.

Ans:

```
kubectl create namespace test-ns
```

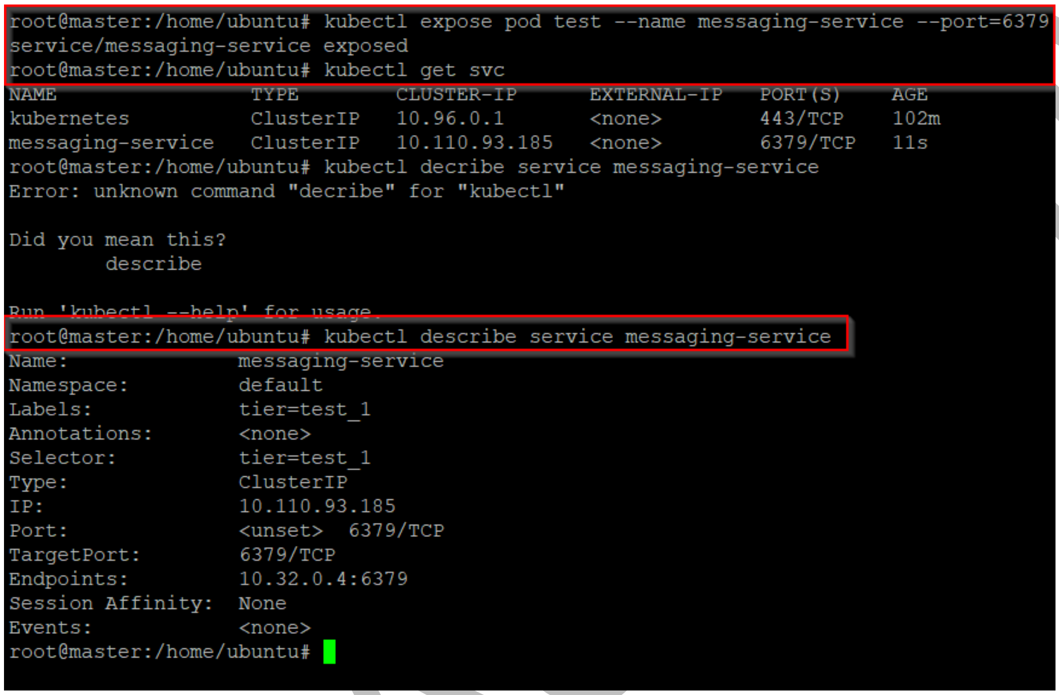


```
root@master:/home/ubuntu# kubectl create namespace test-ns
namespace/test-ns created
root@master:/home/ubuntu# kubectl get namespace
NAME              STATUS   AGE
default           Active   23m
kube-node-lease   Active   23m
kube-public       Active   23m
kube-system       Active   23m
test-ns           Active   10s
root@master:/home/ubuntu#
```

Q4) Create a service **messaging-service** to expose the messaging application within the cluster on **port 6379**.

Ans:

```
kubectl expose pod test --name messaging-service --port=6379
```



```
root@master:/home/ubuntu# kubectl expose pod test --name messaging-service --port=6379
service/messaging-service exposed
root@master:/home/ubuntu# kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes          ClusterIP   10.96.0.1       <none>        443/TCP    102m
messaging-service   ClusterIP   10.110.93.185   <none>        6379/TCP   11s
root@master:/home/ubuntu# kubectl decribe service messaging-service
Error: unknown command "decribe" for "kubectl"

Did you mean this?
        describe

Run 'kubectl --help' for usage.
root@master:/home/ubuntu# kubectl describe service messaging-service
Name:              messaging-service
Namespace:         default
Labels:            tier=test_1
Annotations:       <none>
Selector:          tier=test_1
Type:              ClusterIP
IP:                10.110.93.185
Port:              <unset>  6379/TCP
TargetPort:        6379/TCP
Endpoints:         10.32.0.4:6379
Session Affinity:  None
Events:            <none>
root@master:/home/ubuntu#
```

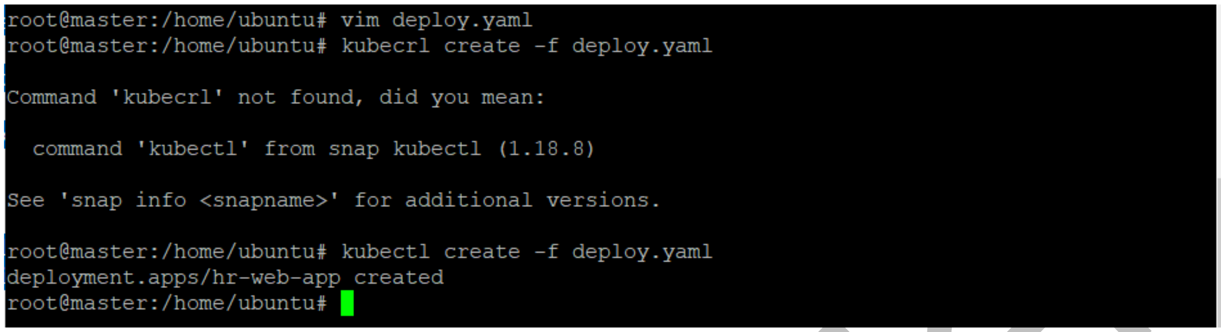
Q5) Create a deployment named **hr-web-app** using the image **nginx** with **2 replicas**.

Ans:

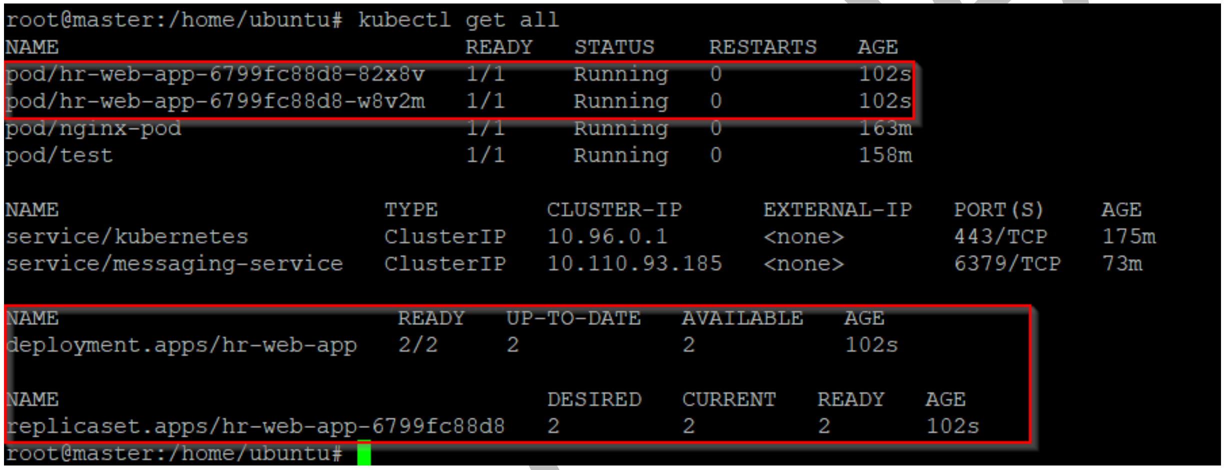
```
vim deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hr-web-app
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```

```
      app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```



```
root@master:/home/ubuntu# vim deploy.yaml
root@master:/home/ubuntu# kubecrl create -f deploy.yaml

Command 'kubecrl' not found, did you mean:

  command 'kubectl' from snap kubectl (1.18.8)

See 'snap info <snapname>' for additional versions.

root@master:/home/ubuntu# kubectl create -f deploy.yaml
deployment.apps/hr-web-app created
root@master:/home/ubuntu#
```



```
root@master:/home/ubuntu# kubectl get all
NAME                              READY   STATUS    RESTARTS   AGE
pod/hr-web-app-6799fc88d8-82x8v   1/1     Running   0          102s
pod/hr-web-app-6799fc88d8-w8v2m   1/1     Running   0          102s
pod/nginx-pod                     1/1     Running   0          163m
pod/test                          1/1     Running   0          158m

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes            ClusterIP   10.96.0.1       <none>        443/TCP    175m
service/messaging-service     ClusterIP   10.110.93.185   <none>        6379/TCP   73m

NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/hr-web-app   2/2     2            2           102s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/hr-web-app-6799fc88d8   2         2         2       102s
root@master:/home/ubuntu#
```

Q6) Create the nginx pod and execute the simple shell on the pod**.**

Ans: // creating a pod

```
kubectl run nginx --image=nginx
```

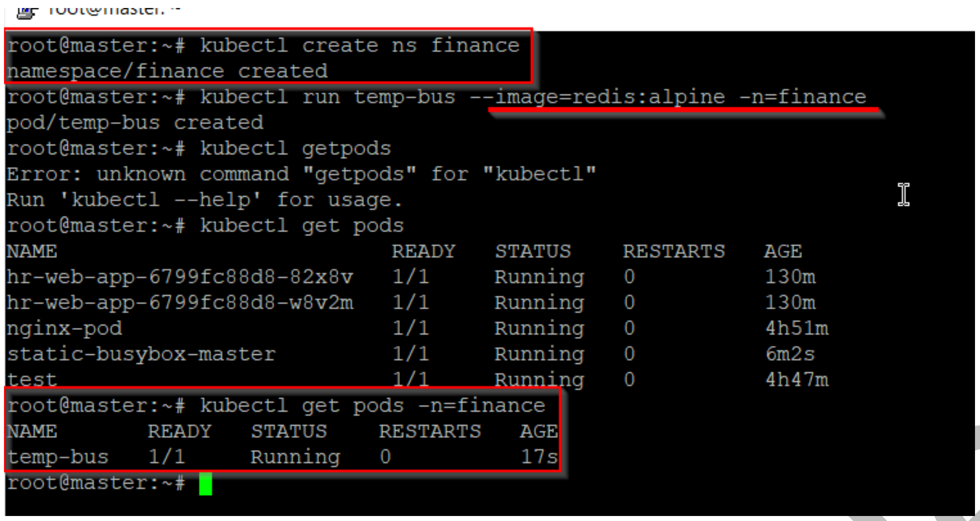
// exec into the pod

```
kubectl exec -it nginx /bin/sh
```

Q7) Create a POD in the **finance namespace** named **temp-bus** with the image **redis:alpine**.

Ans:

```
kubectl create ns finance
kubectl run temp-bus --image=redis:alpine -n=finance
kubectl get pods -n=finance
```

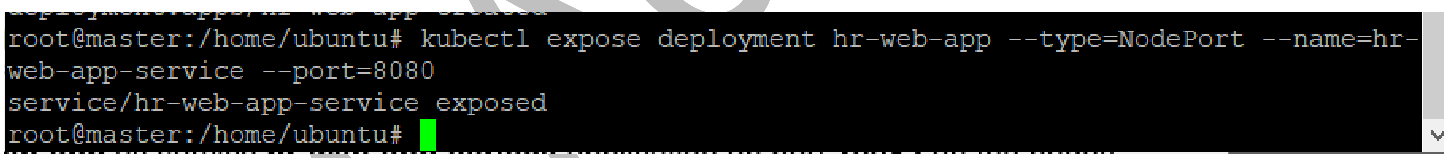


```
root@master:~# kubectl create ns finance
namespace/finance created
root@master:~# kubectl run temp-bus --image=redis:alpine -n=finance
pod/temp-bus created
root@master:~# kubectl getpods
Error: unknown command "getpods" for "kubectl"
Run 'kubectl --help' for usage.
root@master:~# kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
hr-web-app-6799fc88d8-82x8v   1/1     Running   0          130m
hr-web-app-6799fc88d8-w8v2m   1/1     Running   0          130m
nginx-pod                     1/1     Running   0          4h51m
static-busybox-master         1/1     Running   0          6m2s
test                          1/1     Running   0          4h47m
root@master:~# kubectl get pods -n=finance
NAME       READY   STATUS    RESTARTS   AGE
temp-bus   1/1     Running   0          17s
root@master:~#
```

Q8) Expose the **hr-web-app** as service **hr-web-app-service** application on port **30082** on the nodes on the cluster on port **8080**.

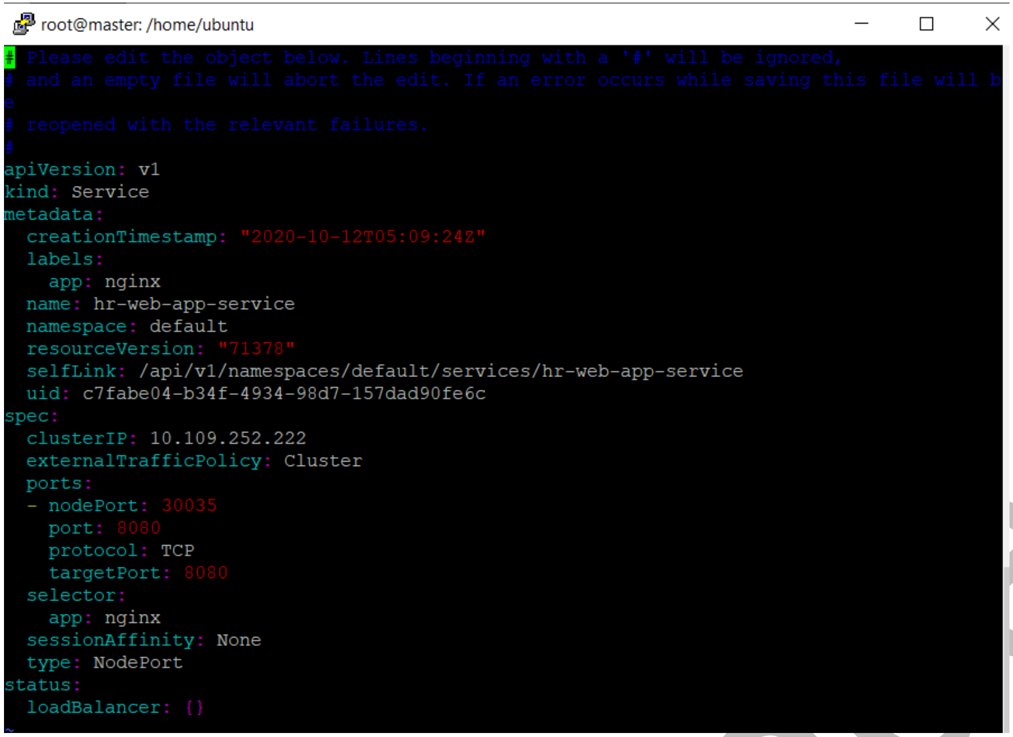
Ans:

```
kubectl expose deployment hr-web-app --type=NodePort --name=hr-web-app-service --port=8080
kubectl edit service/hr-web-app
```



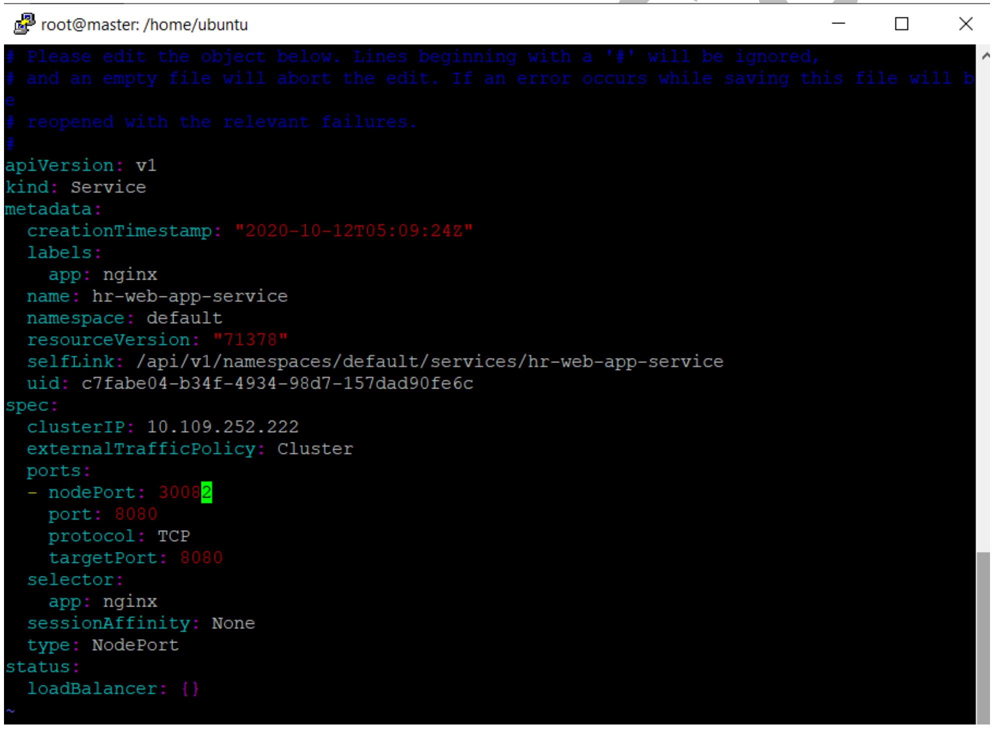
```
root@master:/home/ubuntu# kubectl expose deployment hr-web-app --type=NodePort --name=hr-
web-app-service --port=8080
service/hr-web-app-service exposed
root@master:/home/ubuntu#
```

Before



```
root@master: /home/ubuntu
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will b
e
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-10-12T05:09:24Z"
  labels:
    app: nginx
  name: hr-web-app-service
  namespace: default
  resourceVersion: "71378"
  selfLink: /api/v1/namespaces/default/services/hr-web-app-service
  uid: c7fabe04-b34f-4934-98d7-157dad90fe6c
spec:
  clusterIP: 10.109.252.222
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 30035
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
~
```

After



```
root@master: /home/ubuntu
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will b
e
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-10-12T05:09:24Z"
  labels:
    app: nginx
  name: hr-web-app-service
  namespace: default
  resourceVersion: "71378"
  selfLink: /api/v1/namespaces/default/services/hr-web-app-service
  uid: c7fabe04-b34f-4934-98d7-157dad90fe6c
spec:
  clusterIP: 10.109.252.222
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 30082
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: nginx
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
~
```

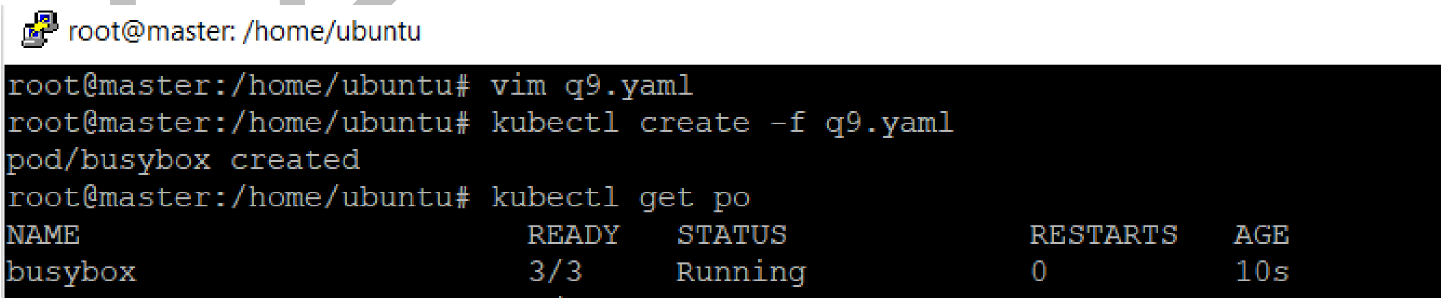
Q9) Create a Pod with three busy box containers with commands “ls; sleep 3600;”, “echo Hello World; sleep 3600;” and “echo this is the third container; sleep 3600” respectively and check the status.

Ans:

```
vim q9.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
  name: busybox
spec:
  containers:
  - args:
    - bin/sh
    - -c
    - ls; sleep 3600
    image: busybox
    name: busybox1

  - args:
    - bin/sh
    - -c
    - echo Hello world; sleep 3600
    image: busybox
    name: busybox2

  - args:
    - bin/sh
    - -c
    - echo this is third container; sleep 3600
    image: busybox
    name: busybox3
```

```
kubectl create -f q9.yaml

kubectl get pod
```

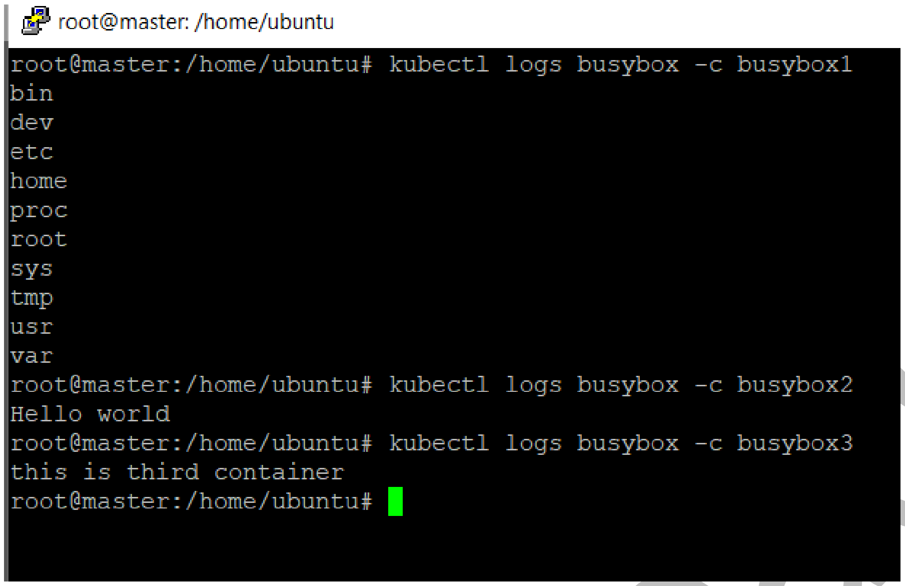


```
root@master: /home/ubuntu
root@master:/home/ubuntu# vim q9.yaml
root@master:/home/ubuntu# kubectl create -f q9.yaml
pod/busybox created
root@master:/home/ubuntu# kubectl get po
NAME                        READY   STATUS             RESTARTS   AGE
busybox                     3/3     Running            0          10s
```

Q10) Check the logs of each container in above question.

Ans:

```
kubectl logs busybox -c busybox1
kubectl logs busybox -c busybox2
kubectl logs busybox -c busybox3
```

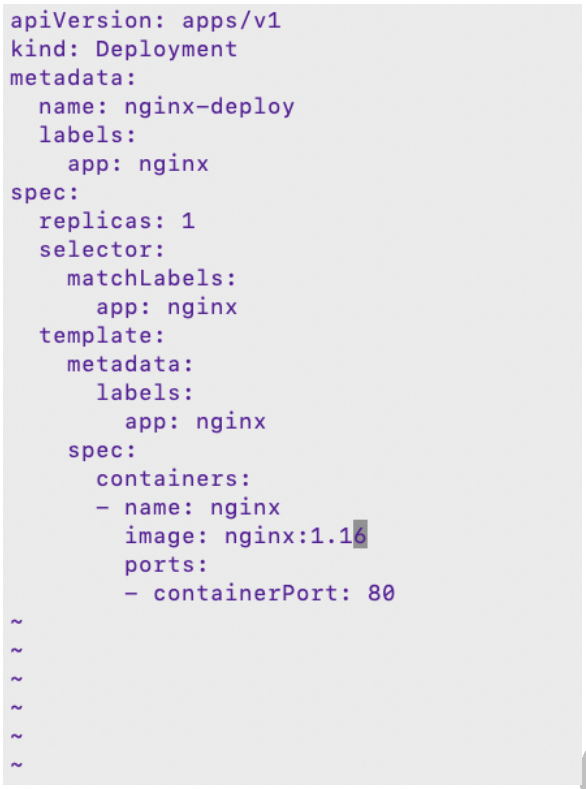


root@master: /home/ubuntu

```
root@master:/home/ubuntu# kubectl logs busybox -c busybox1
bin
dev
etc
home
proc
root
sys
tmp
usr
var
root@master:/home/ubuntu# kubectl logs busybox -c busybox2
Hello world
root@master:/home/ubuntu# kubectl logs busybox -c busybox3
this is third container
root@master:/home/ubuntu#
```

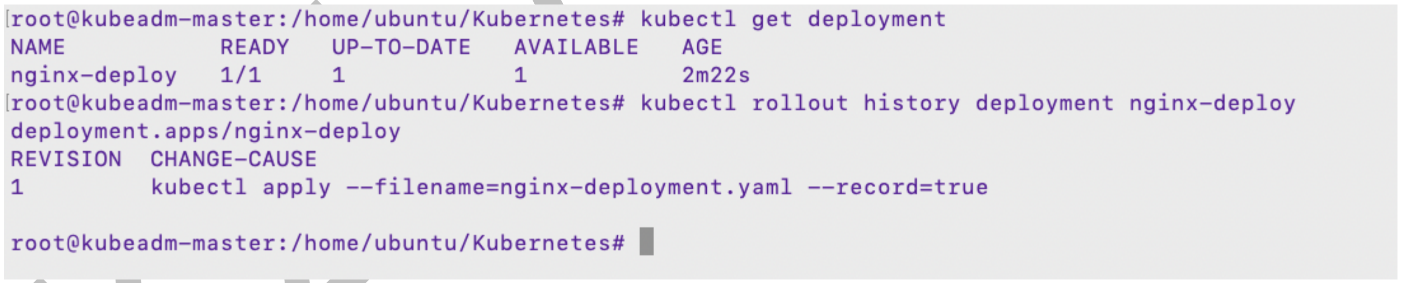
Q11) Create a new deployment called **nginx-deploy**, with image **nginx:1.16** and **1 replica**. Record the version. Next upgrade the deployment to version **1.17** using rolling update. Make sure that the version upgrade is recorded in the resource annotation.

Ans:



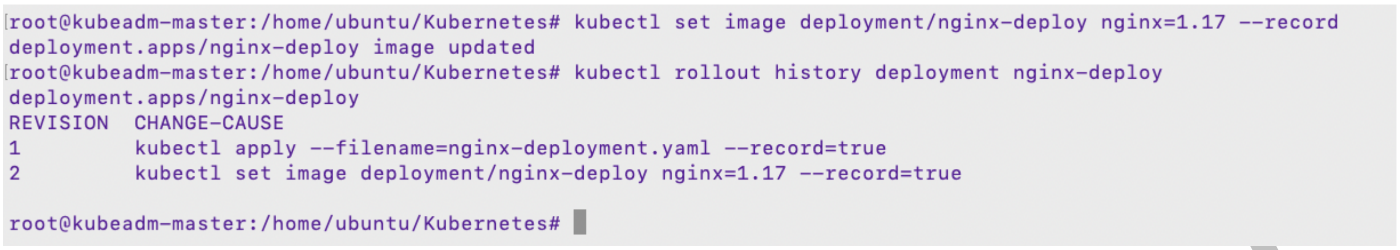
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 80
~
~
~
~
~
~
```

```
vim nginx-deployment.yaml
kubectl apply -f nginx-deployment.yaml --record
kubectl get deployment
kubectl rollout history deployment nginx-deploy
```

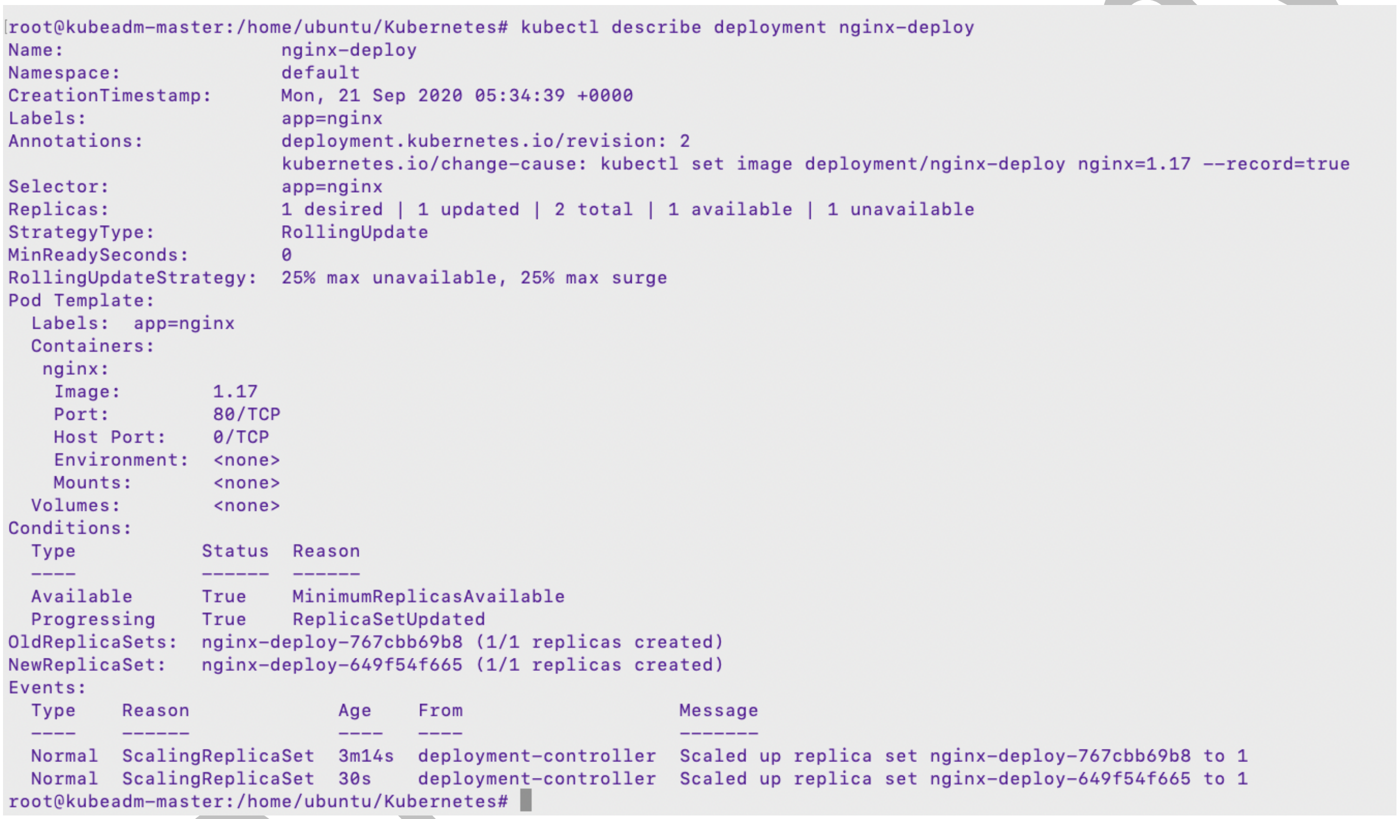


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deploy   1/1     1            1           2m22s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl rollout history deployment nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=nginx-deployment.yaml --record=true

root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
kubectl set image deployment/nginx-deploy nginx=1.17 --record
kubectl rollout history deployment nginx-deploy
```


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl set image deployment/nginx-deploy nginx=1.17 --record
deployment.apps/nginx-deploy image updated
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl rollout history deployment nginx-deploy
deployment.apps/nginx-deploy
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=nginx-deployment.yaml --record=true
2         kubectl set image deployment/nginx-deploy nginx=1.17 --record=true

root@kubeadm-master:/home/ubuntu/Kubernetes# 
```

```
kubectl describe deployment nginx-deploy
```


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe deployment nginx-deploy
Name:                   nginx-deploy
Namespace:              default
CreationTimestamp:      Mon, 21 Sep 2020 05:34:39 +0000
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 2
                        kubernetes.io/change-cause: kubectl set image deployment/nginx-deploy nginx=1.17 --record=true
Selector:               app=nginx
Replicas:               1 desired | 1 updated | 2 total | 1 available | 1 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        1.17
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets:  nginx-deploy-767cbb69b8 (1/1 replicas created)
NewReplicaSet:   nginx-deploy-649f54f665 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m14s  deployment-controller  Scaled up replica set nginx-deploy-767cbb69b8 to 1
  Normal  ScalingReplicaSet  30s    deployment-controller  Scaled up replica set nginx-deploy-649f54f665 to 1
root@kubeadm-master:/home/ubuntu/Kubernetes# 
```

Q12) Create a Pod with main container busybox and which executes this "while true; do echo 'Hi I am from Main container' >> /var/log/index.html; sleep 5; done" and with sidecar container with nginx image which exposes on port 80.

Use emptyDir Volume and mount this volume on path /var/log for busybox and on path /usr/share/nginx/html for nginx container. Verify both containers are running**.**

Ans:

```
vim sidecar.yaml
```

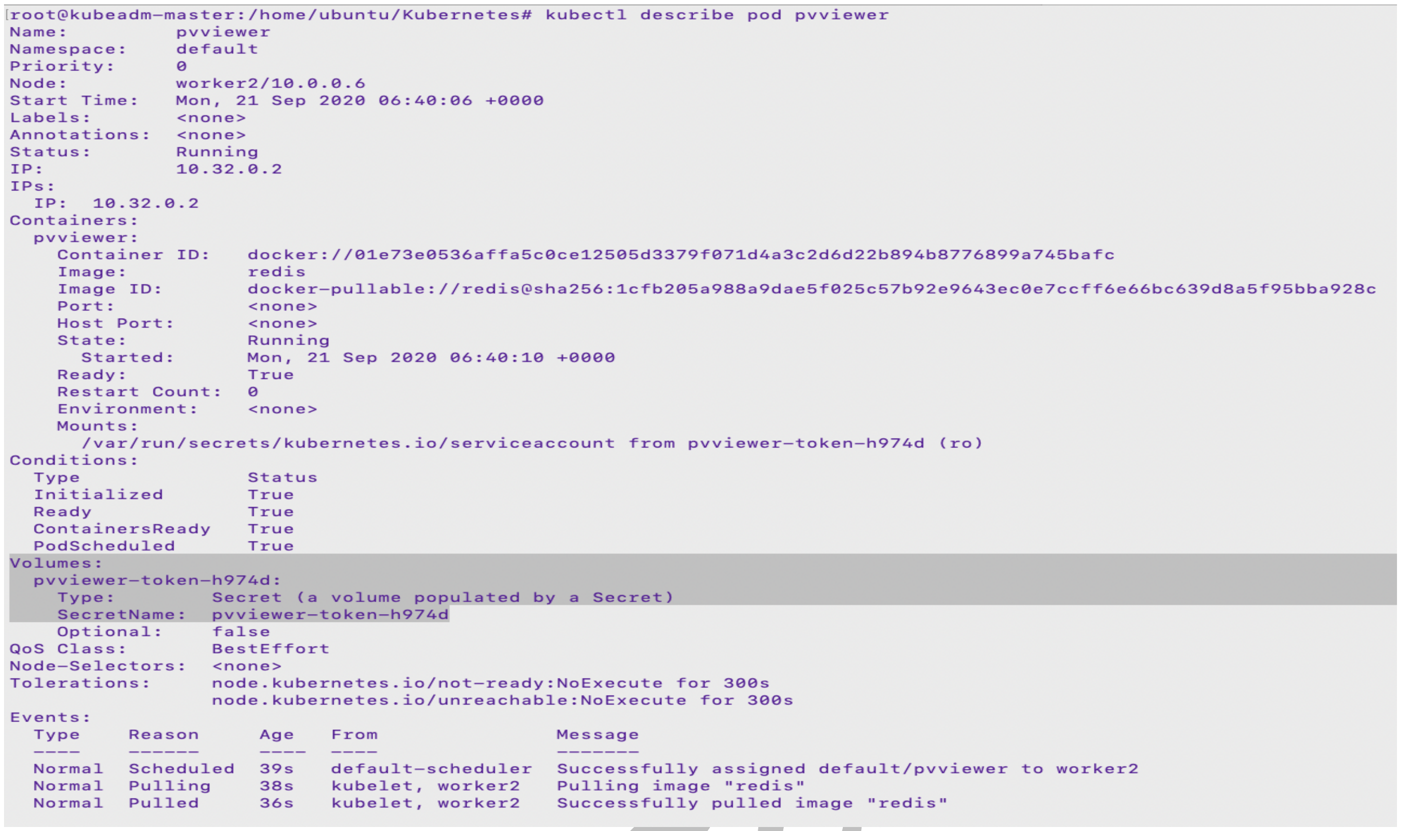
```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: multi-cont-pod
  name: multi-cont-pod
spec:
  volumes:
  - name: var-logs
    emptyDir: {}
  containers:
  - image: busybox
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo 'Hi I am from Main container' >> /var/log/index.html; sleep 5;done"]
    name: main-container

    volumeMounts:
    - name: var-logs
      mountPath: /var/log
  - image: nginx
    name: sidecar-container
    ports:
      - containerPort: 80
    volumeMounts:
    - name: var-logs
      mountPath: /usr/share/nginx/html
```

```
kubectl create -f sidecar.yaml
```

Q13) Exec into both containers and verify that main.txt exist and query the main.txt from sidecar container with curl localhost.

Ans: // exec into side container

```
kubectl exec -it  multi-cont-pod -c sidecar-container -- sh
cat /usr/share/nginx/html/index.html
curl localhost
```



```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl describe pod pvviewer
Name:         pvviewer
Namespace:    default
Priority:     0
Node:         worker2/10.0.0.6
Start Time:   Mon, 21 Sep 2020 06:40:06 +0000
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.32.0.2
IPs:
  IP:  10.32.0.2
Containers:
  pvviewer:
    Container ID:   docker://01e73e0536affa5c0ce12505d3379f071d4a3c2d6d22b894b8776899a745bafc
    Image:          redis
    Image ID:       docker-pullable://redis@sha256:1cfb205a988a9dae5f025c57b92e9643ec0e7ccff6e66bc639d8a5f95bba928c
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 21 Sep 2020 06:40:10 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from pvviewer-token-h974d (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  pvviewer-token-h974d:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  pvviewer-token-h974d
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  39s   default-scheduler  Successfully assigned default/pvviewer to worker2
  Normal  Pulling    38s   kubelet, worker2   Pulling image "redis"
  Normal  Pulled     36s   kubelet, worker2   Successfully pulled image "redis"
```

Q14) Create a pod called **multi-pod** with two containers.

Container 1, name: alpha, image: nginx
Container 2: beta, image: busybox, command sleep 4800

Environment Variables: Container1 → name: alpha
Environment Variables: Container2 → name: beta

Ans:

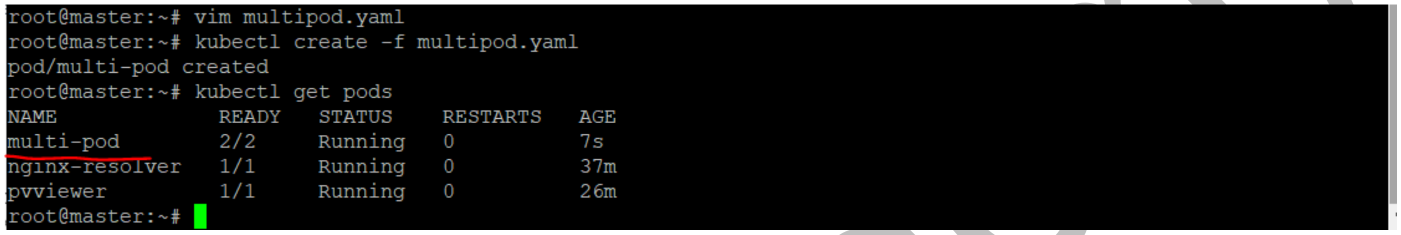
kubectl run --generator=run-pod/v1 alpha --image=nginx --dry-run -o yaml > multi-pod.yaml

#Edit the yaml file

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
spec:
  containers:
  - image: nginx
    name: alpha
    env:
```

```
    - name: name
      value: alpha
  - image: busybox
    name: beta
    env:
    - name: name
      value: beta
    command: ["sleep","4800"]
```

```
kubectl create -f multipod.yaml
```



```
root@master:~# vim multipod.yaml
root@master:~# kubectl create -f multipod.yaml
pod/multi-pod created
root@master:~# kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
multi-pod       2/2     Running   0          7s
nginx-resolver  1/1     Running   0          37m
pvviewer        1/1     Running   0          26m
root@master:~#
```

Q15) Create a Pod called **non-root-pod** , image: **redis:alpine**

runAsUser: 1000

fsGroup: 2000

Ans:

```
vim non-root-pod.yaml

kubectl create -f non-root-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name:  non-root-pod
spec:
  securityContext:
    runAsUser:  1000
    fsGroup:  2000
  containers:
  -  name:  non-root-pod
     image:  redis:alpine
```

Q16) Taint the worker node to be **Unschedulable**. Once done, create a pod called **dev-redis**, image redis:alpine to ensure workloads are not scheduled to this worker node. Finally, create a new pod called **prod-redis** and image redis:alpine with toleration to be scheduled on node01.

key:env_type, value:production, operator: Equal and effect:NoSchedule

Ans:

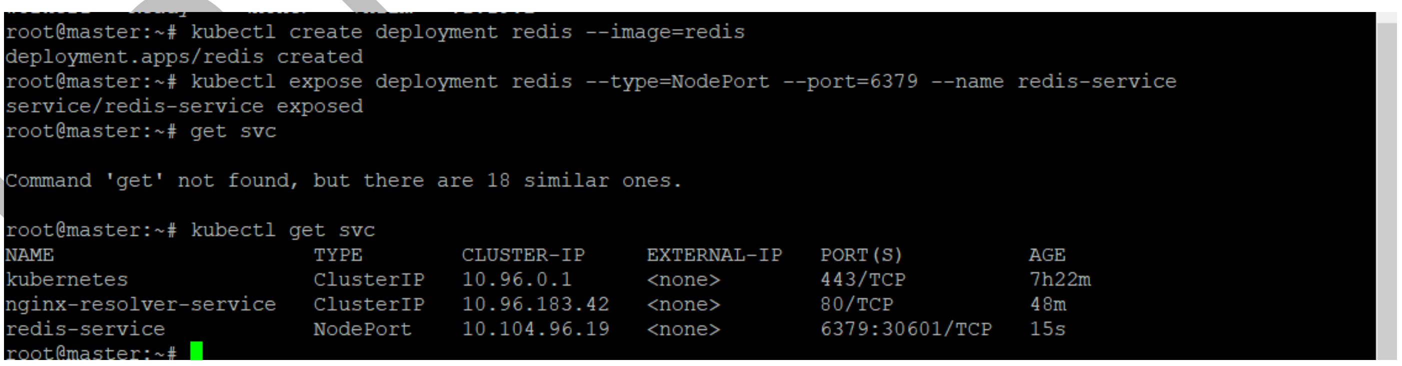
```
kubectl get nodes
kubectl taint node node01 env_type=production:NoSchedule
kubectl describe nodes node01 | grep -i taint
kubectl run dev-redis --generator=run-pod/v1 --image=redis:alpine
kubectl create -f prod-redis.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: prod-redis
spec:
  containers:
  - name:  prod-redis
    image:  redis:alpine
  tolerations:
  - effect: Noschedule
    key: env_type
    operator: Equal
    value: prodcution
```

Q17) Create the deployment redis with image=redis and expose it with "NodePort" service redis-service.

Ans:

```
kubectl create deployment redis --image=redis
kubectl expose deployment redis --type=NodePort --port=6379 --name redis-service
```

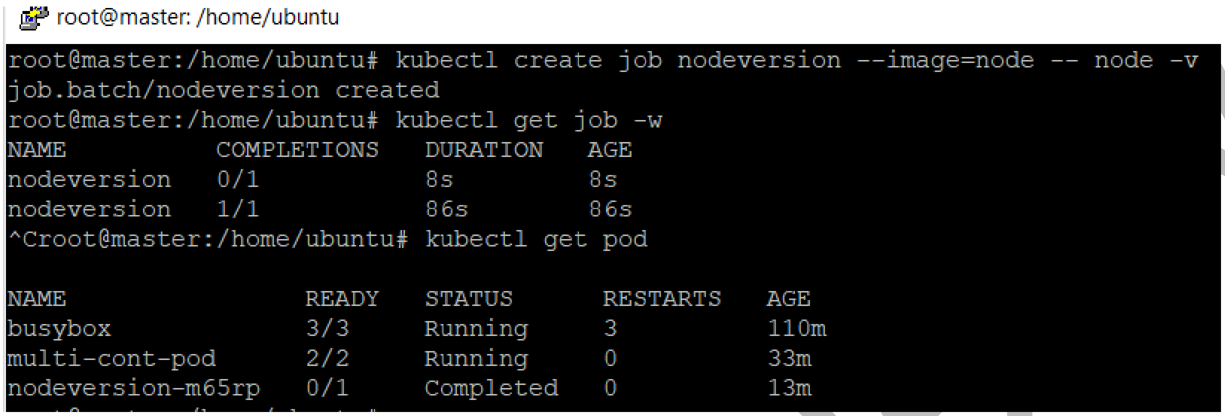


```
root@master:~# kubectl create deployment redis --image=redis
deployment.apps/redis created
root@master:~# kubectl expose deployment redis --type=NodePort --port=6379 --name redis-service
service/redis-service exposed
root@master:~# get svc

Command 'get' not found, but there are 18 similar ones.

root@master:~# kubectl get svc
NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes               ClusterIP   10.96.0.1       <none>        443/TCP          7h22m
nginx-resolver-service   ClusterIP   10.96.183.42    <none>        80/TCP           48m
redis-service            NodePort    10.104.96.19    <none>        6379:30601/TCP   15s
root@master:~#
```

Q18) Create a Job with an image node which prints node version and also verifies there is a pod created for this job.

Ans:

```
kubectl create job nodeversion --image=node -- node -v
kubectl get job -w
kubectl get pod
```



```
root@master: /home/ubuntu
root@master:/home/ubuntu# kubectl create job nodeversion --image=node -- node -v
job.batch/nodeversion created
root@master:/home/ubuntu# kubectl get job -w
NAME          COMPLETIONS   DURATION   AGE
nodeversion   0/1           8s         8s
nodeversion   1/1           86s        86s
^Croot@master:/home/ubuntu# kubectl get pod

NAME                READY   STATUS      RESTARTS   AGE
busybox             3/3     Running     3          110m
multi-cont-pod      2/2     Running     0          33m
nodeversion-m65rp   0/1     Completed   0          13m
```

Q19) Get the logs of the job just created.

```
kubectl logs nodeversion-m65rp
```

Q20) Create a Cronjob with busybox image that prints date and hello from kubernetes cluster message for every minute.

```
kubectl create cronjob date-job --image=busybox --schedule="*/1 * * * *" -- bin/sh -c "date; echo Hello from kubernetes cluster"
```

Q21) Verify that CronJob creating a separate job and pods for every minute to run and verify the logs of the pod.
Ans:

```
kubectl get job
kubectl get pod
kubectl logs date-job-<jobid>-<pod>
```

Q22) Delete the CronJob and verify all the associated jobs and pods are also deleted.

Ans:

```
kubectl delete cj date-job
```

Verify//

```
kubectl get pod
kubectl get job
```

Q23) Create and configure the service front-end-service so its accessible through ClusterIP and routes to the existing pod named front-end.

Ans:

```
kubectl expose pod front-end --name front-end-service --port=80
```

Q24) Scale the deployment webserver to 3 pods(replicas).

Ans:

```
kubectl scale --replicas=3 deployment/webserver
```

Q25) Create a persistent volume with name **app-data** , of capacity **1 Gi** and access mode **ReadWriteOnce**. The type of volume is **hostPath** and its location is **/srv/app/data**.

Ans:

```
vim pv.yaml

kubectl get pv
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/srv/app/data"
```

```
kubectl create -f pv.yaml
```

Q26) Create a PersistentVolumeClaim of 1Gi storage with name task-pv-claim and access mode ReadWriteOnce and verify status is Bound.

Ans:

```
vim pvc.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
spec:
```

```
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 1Gi
```

```
kubectl create -f pvc.yaml
```

Q27) Create a configmap called myconfigmap with literal value appname=myapp.

Ans:

```
kubectl create cm myconfigmap --from-literal=appname=myapp
```

Q28) Verify the configmap we just created has this data.

Ans:

```
kubectl describe cm myconfigmap
```

Q29) Create a file called config.txt with two values key1=value1 and key2=value2 and Create a configmap named keyvalcfgmap and read data from the file config.txt.

Ans:

```
cat >> config.txt << EOF
key1=value1
key2=value2
EOF


kubectl create cm keyvalcfgmap --from-file=config.txt
kubectl get cm keyvalcfgmap
```

Q30) Create an nginx pod and load environment values from the above configmap keyvalcfgmap and exec into the pod and verify the environment variables.

Ans:

```
vim nginx-pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    envFrom:
    - configMapRef:
        name: keyvalcfgmap
```

```
kubectl create -f nginx-pod.yml
```

Q31) Create a secret mysecret with values user=myuser and password=mypassword.

Ans:

```
kubectl create secret generic my-secret --from-literal=username=user --from-literal=password=mypassword
```

Q32) Create an nginx pod which reads username as the environment variable.

Ans:

```
vim nginx-secret.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    env:
    - name: USER_NAME
      valueFrom:
        secretKeyRef:
          name: my-secret
          key: username
```

```
kubectl create -f nginx-secret.yml
```