Q. Reconfigure the existing deployment front-end and add a port specification named http , exposing port 80/tcp of the exising continaer nginx.

Q. Create a New service named front-end-svc exposing the container port http

Q. Configure new service to also expose the individual pods via a NodePort
  on the nodes on which theyu are scheduled.

==========

Q. Create a pod with nginx image, named as web in dev-test namespace
SOLU
====
```
$ kubectl  create  ns  dev-test
$ kubectl  run  web  --image=nginx  --restart=Never  -n dev-test
```

==========

Q. show the pods that are part of namespace XYZ , show services that is part of namespace ABC
SOLU
====
```
$ kubectl get  po  -n XYZ
$ kubectl get  svc -n ABC
```

==========

```
kube3a k top po -n kube-system | head -3
NAME                              CPU(cores)   MEMORY(bytes)
coredns-86c58d9df4-562mf          1m           7Mi
coredns-86c58d9df4-zl4mj          1m           7Mi
```

List the top 1 pod with the highest CPU usage
```
     kubectl top po -n kube-system  | sort -nr  -k2 | head  -1
                                                 ^^<<== -k is colum number , so colum 2 is the CPU
```

List the top 1 pod with the highest MEM usage
```
     kubectl top po -n kube-system  | sort -nr  -k3 | head -1
                                                ^^<<== -k is column ,  Column 3 is the Memory
```

==========

1. Create a static pod on a node and put the pod specification file in /etc/kubernetes/manifests.
SOLU
=====
Refer to https://kubernetes.io/docs/tasks/administer-cluster/static-pod/ , and use the static-web.yaml
root@kube3b:/etc/kubernetes/manifests# pwd
/etc/kubernetes/manifests
root@kube3b:/etc/kubernetes/manifests# cat static-web.yaml
apiVersion: v1
kind: Pod
metadata:
 name: static-web
 labels:
   role: myrole
spec:
 containers:
   - name: web
     image: nginx
     ports:
       - name: web
         containerPort: 80
         protocol: TCP
root@kube3b:/etc/kubernetes/manifests#

Refer to : https://medium.com/@sonasingh46/static-pod-in-kubernetes-e3854507655f
vi /etc/systemd/system/kubelet.service
[Service]
 ExecStart=/home/kubernetes/bin/kubelet $KUBELET_OPTS   \
           --pod-manifest-path=/etc/kubernetes/manifests

systemctl daemon-reload ; kubectl get po ; if no pod then
systemctl restart kubelet

==========

2. Create a deployment nginx-dns and expose it as service nginx-dns.
 Create busybox pod , Use nslookup to get dns result for service & pod and store the values into a file.
SOLU
====
kube3a k run nginx-dns --image=nginx --port=80

```
     deployment.apps/nginx-dns created
kube3a k expose deployment nginx-dns  --port=80 --target-port=80
     service/nginx-dns exposed
kube3a

by default  nginx assings  label as run=deploymet|pod_name>

kube3a k get  deploy,po,svc  -l run=nginx-dns -o wide
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS   IMAGES   SELECTOR
deployment.extensions/nginx-dns   1/1     1            1           4m44s   nginx-
dns    nginx    run=nginx-dns

NAME                                READY   STATUS    RESTARTS   AGE     IP
NODE                NOMINATED NODE   READINESS GATES
pod/nginx-dns-6b7f858b9b-hzcvc   1/1     Running   0          4m44s   10.244.2.153
kube3c.home.local   <none>           <none>

^^^^^^^^^^<<== we need to resolve this for pod.dns

NAME                TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
SELECTOR
service/nginx-dns   ClusterIP   10.98.27.246   <none>        80/TCP     3m47s
run=nginx-dns
kube3a                          ^^^^^^^^^^<<== we need to resolve this for
service.dns


kube3a k run bbox --image=busybox:1.28 --restart=Never -- /bin/sh -c 'sleep 360000'
     pod/bbox created

Resolve pod IP  :
kube3a k exec bbox -- nslookup 10.244.2.153  > /tmp/aa
kube3a
Edit /tmp/aa and delete everything except :
     10-244-2-153.nginx-dns.default.svc.cluster.local

kube3a sudo cp /tmp/aa /opt/TTT/pod.dns
kube3a cat /opt/TTT/pod.dns
     10-244-2-153.nginx-dns.default.svc.cluster.local


Next to resolve service for pod :
kube3a k exec bbox -- nslookup nginx-dns  > /tmp/bb
kube3a
```

```
Edit /tmp/bb and ensure you ONLY have :
kube3a cat /tmp/bb
     nginx-dns.default.svc.cluster.local
kube3a

kube3a sudo cp /tmp/bb  /opt/TTT/service.dns
kube3a cat /opt/TTT/service.dns
     nginx-dns.default.svc.cluster.local
kube3a

=========================================================

3. Create a init container which shouldn't start if a file is not found.
SOLU
====

I created  :
$ k run bbox --image=busybox --restart=Never --dry-run -o yaml -- /bin/sh  -c 'if [
! -f /opt/XX/xx]
>then
>exit 0
>else
>sleep 360000000
>fi'   > bbox12.yaml

Once I had above then I added entries for volume: and volumeMounts: for emptyDir,
then

dharmin@kube3a:~$ cat bbox12.yaml
apiVersion: v1
kind: Pod
metadata:
 creationTimestamp: null
 labels:
   run: bbox
 name: bbox
spec:
 volumes:                <<== U need emptyDir: or hostpath:
 - name: cache-volume    <<==
   emptyDir: {}          <<==
 containers:
 - args:
   - /bin/sh
   - -c
   - |-
```

```
     if [ ! -f /opt/XX/xx ]
     then
     exit 0
     else
     sleep 36000000
     fi
   image: busybox
   name: bbox
   volumeMounts:
   - mountPath: /opt/XX  <<==  the initcontainer and the actual container shares
same dir
     name: cache-volume
 initContainers:
 - args:
   - /bin/sh
   - -c
   - 'touch /opt/XX/xx'
   image: busybox
   name: bbox1
   volumeMounts:
   - mountPath: /opt/XX  <<== the initcontainer and the actual container shares
same dir
     name: cache-volume

#
##  dnsPolicy: ClusterFirst
##  restartPolicy: Never


kube3a k create -f bbox12.yaml
     pod/bbox created

kube3a k get po --watch
NAME   READY   STATUS     RESTARTS   AGE
bbox   0/1     Init:0/1   0          12s
bbox   0/1   PodInitializing   0     16s
bbox   1/1   Running   0     18s
bbox   0/1   Completed   0     19s
bbox   1/1   Running   1     21s
bbox   0/1   Completed   1     22s
^

kube3a k logs bbox
     FILE PRESENT BOSSS
```

```
================================================================================
4. Bootstrap a kubernetes node.
https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-
bootstrapping/

================================================================================

5. Use node selector to assign a pod to particular node.
SOLU
====
Label a Node :
$ kubectl label no kube3c   node=kube3c

Create  alpine  pod , and assign to a Node labeled node=kube3c :
kube3a k run alpine --image=alpine --restart=Never --dry-run -o yaml --command
sleep 36000000 > alpine-nodeselector.yaml

kube3a cat alpine-nodeselector.yaml
apiVersion: v1
kind: Pod
metadata:
 creationTimestamp: null
 labels:
   run: alpine
 name: alpine
spec:
 containers:
 - command:
   - sleep
   - "36000000"
   image: alpine
   name: alpine
   resources: {}
 dnsPolicy: ClusterFirst
 restartPolicy: Never
 nodeSelector:        <<== edit file and add this  manually
   node: kube3c       <<== add this  and add this manually
kube3a

kube3a k create -f  alpine-nodeselector.yaml
     pod/alpine created

kube3a k get po alpine -o wide
NAME     READY   STATUS              RESTARTS   AGE   IP       NODE
NOMINATED NODE   READINESS GATES
```

```
alpine   0/1     ContainerCreating   0          13s   <none>   kube3c.home.local
<none>           <none>
kube3a
```

==========

6. Label a node and it shouldn't be allowed to execute/schedule any pods.

SOLU

====

```
At present :
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <none>
Taints:             <none>
kube3a

kube3a  k taint node kube3c aa=bb:NoSchedule
kube3a  k describe no | egrep -i taint
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <none>
Taints:             aa=bb:NoSchedule
kube3a

kube3a  k get po --all-namespaces -o wide | egrep kube3c | wc -l
     2
```

Next deploy 20 nginx pods and see if any gets deployed on kube3c :

```
kube3a  k get po --all-namespaces  -o wide | egrep -i kube3c | wc -l
     2
```

AS per above we still have 2 pods that are runing on kube3c , as those are part of daemonset

==========

7. Create a deployment name nginx-app with 3 replicas use nginx:1.9.1, do a rolling update to nginx:1.13.5 and then undo it.

  New image was asked to use for rolling update.

SOLU

====

```
kube3a  k run nginx-app --image=nginx:1.9.1 --replicas=3
     deployment.apps/nginx-app created

kube3a  k rollout status deploy nginx-app
```

```
     deployment "nginx-app" successfully rolled out

kube3a  k describe deploy nginx-app | egrep -i image:
   Image:        nginx:1.9.1
kube3a

Next update the nginx to ver 1.13.5
kube3a  k set image deployment nginx-app   nginx-app=nginx:1.13.5
deployment.extensions/nginx-app image updated
kube3a  k rollout status deploy nginx-app
Waiting for deployment "nginx-app" rollout to finish: 1 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 1 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 1 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 2 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 2 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 2 out of 3 new replicas have
been updated...
Waiting for deployment "nginx-app" rollout to finish: 1 old replicas are pending
termination...
Waiting for deployment "nginx-app" rollout to finish: 1 old replicas are pending
termination...
deployment "nginx-app" successfully rolled out
kube3a

kube3a  k describe deployment nginx-app | egrep -i image:
   Image:        nginx:1.13.5
kube3a

Undo the upgraqde to version 1 ( previous ) :
kube3a  k rollout  undo deployment nginx-app --to-revision=1
     deployment.extensions/nginx-app rolled back

kube3a  k describe deploy nginx-app | egrep -i image:
   Image:        nginx:1.9.1
kube3a

================================================================================
```

8. Node is in not ready status. Find the issue and fix it.
SOLU
=====
ssh onto the worker Node , and run :
$ sudo systemctl status -l kubelet

If the kubelete service is not runing then :
$ sudo systemctl enable kubelet
$ sudo systemctl restart kubelet

Finally run below to ensure kubelet service is runing fine :
$ sudo systemctl status -l kubelet

==========

9. Find all pods which are using same service in a namespace and store the value into a file.
SOLU
======
Create a ns called boss :
kube3a k create ns boss
     namespace/boss created

kube3a k get ns boss
NAME   STATUS   AGE
boss   Active   10s
kube3a

deploy busybox pods and expose port 80 and create service in ns boss.
kube3a k get po -n boss
NAME   READY   STATUS   RESTARTS   AGE
b   1/1   Running   0   2m27s
b1   1/1   Running   0   2m22s
b2   1/1   Running   0   2m15s
b3   1/1   Running   0   109s
b4   1/1   Running   0   103s
b5   1/1   Running   0   98s
b6   1/1   Running   0   86s
kube3a

above is total 8 , but then we ignore the header "NAME READY STATUS RESTARTS AGE" , si its 7
$ sudo echo 7 > /opt/file/X

```
kube3a  k get svc -n boss
NAME   TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
b3     ClusterIP   10.96.172.82     <none>        80/TCP    8m
b4     ClusterIP   10.107.213.233   <none>        80/TCP    7m54s
b5     ClusterIP   10.98.80.82      <none>        80/TCP    7m49s
b6     ClusterIP   10.102.82.0      <none>        80/TCP    7m37s
kube3a  k get svc -n boss | wc -l
     5
```

By ignoring headers , we have 4 :

```
$ sudo echo 4 > /opt/file/XX
```

==========

10. Find all pods with label cpu=loader and sort them by consuption and store the value into a file.

SOLU

====

I have create couple of pods , all have label cpu=loader, lets see which has High CPU Usage
and which consumes more Memory.,

```
kube3a k top po -l cpu=loader
NAME   CPU(cores)   MEMORY(bytes)
a1     0m           2Mi
a2     1m           4Mi
a3     6m           14Mi
a4     1m           1Mi
a5     0m           0Mi
a6     0m           0Mi
kube3a

echo "a3" >  high.cpu
echo "a3" > high.mem


Delete all the above pods that have label cpu: loader
kube3a k delete po -l cpu=loader
pod "a1" deleted
pod "a2" deleted
pod "a3" deleted
pod "a4" deleted
pod "a5" deleted
pod "a6" deleted
```

kube3a

==========

11. Find all persistent volumes in a namespace and sort them by capacity and store the value into a file.
SOLU
=====

```
kube3a k get pv --sort-by=.spec.capacity.storage
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
pv-1g       1Gi        RWO            Retain           Available
103s
myvolume    10Gi       RWO,RWX        Retain           Available           normal
4m5s
myvolume1   15Gi       RWO,RWX        Retain           Available           normal
3m11s
pv-1f       100Gi      RWO            Retain           Available
57s
kube3a
```

==========

12. Create a pod with multiple containers including image of redis, nginx , memcached, consul, busybox and alpine.
SOLU
=====
NOTE : ONLY busybox and alpine requires "--command sleep 3600000" , and rest does NOT

```
$ k run pod-all --image=alpine --restart=Never --dry-run   -o yaml --command sleep 3600000 > pod-all.yaml

kube3a cat  pod-all.yaml
apiVersion: v1
kind: Pod
metadata:
 creationTimestamp: null
 labels:
   run: pod-all
 name: pod-all
```

```
spec:
 containers:
 - command:
   - sleep
   - "3600000"
   image: alpine
   name: p1
#
 - command:
   - sleep
   - "3600000"
   image: busybox
   name: p2
#
 - image: redis
   name: p3
#
 - image: memcached
   name: p4
#
 - image: nginx
   name: p5
#
 - image: consul
   name: p6
#
 dnsPolicy: ClusterFirst
 restartPolicy: Never
#
kube3a
kube3a k create -f  pod-all.yaml
     pod/pod-all created

After few mins..
kube3a k get po
NAME      READY   STATUS    RESTARTS   AGE
pod-all   6/6     Running   0          4m52s
kube3a
```

==========

13. Create a secret super-secret with credential as bob and create two pods, one which mount secret as volume and another which mount it as env variable.

```
SOLU
=====
kube3a k create secret generic super-secret --from-literal=credential=bob
     secret/super-secret created

kube3a k get secret super-secret
NAME           TYPE     DATA   AGE
super-secret   Opaque   1      76s

kube3a k get secret super-secret --export -o yaml
apiVersion: v1
data:
 credential: Ym9i      <<== echo Ym9i | base64 -d , will give you bob
kind: Secret
metadata:
 creationTimestamp: null
 name: super-secret
 selfLink: /api/v1/namespaces/default/secrets/super-secret
type: Opaque
kube3a

Create environment variable called TOPSECRET and assign the secret value , refer to
:
https://kubernetes.io/docs/concepts/configuration/secret/

$ pod-secret.yaml
apiVersion: v1
kind: Pod
metadata:
 name: secret-env-pod
spec:
 containers:
 - name: mycontainer
   image: redis
   env:
     - name: TOPSECRET
       valueFrom:
         secretKeyRef:
           name: super-secret
           key: credential

In the above secret url , I searched for env: and saw above , which I cp & paste

kube3a k create -f  pod-secret.yaml
     pod/secret-env-pod created
```

```
kube3a

kube3a k get po
NAME              READY   STATUS    RESTARTS   AGE
secret-env-pod    1/1     Running   0          80s
kube3a

kube3a k  exec secret-env-pod -- printenv | egrep TOPSECRET
     TOPSECRET=bob
kube3a


Next we need to copy the secret in a volume :
kube3a cat secret-pod.yaml
kind: Pod
apiVersion: v1
metadata:
 name: secret-pod
 labels:
   name: secret-pod
spec:
 volumes:
 - name: secret-volume
   secret:
     secretName: super-secret
 containers:
 - name: secret-pod
   image: nginx
   volumeMounts:
   - name: secret-volume
     readOnly: true
     mountPath: "/etc/secret-volume"
kube3a

From the secret url , I searched for volume:  , and found above , which I did minor
changes and bingo..

kube3a k create -f  secret-pod.yaml
     pod/secret-pod created
kube3a
kube3a k exec secret-pod -- ls -l /etc/secret-volume
     lrwxrwxrwx 1 root root 17 Mar 19 22:58 credential -> ..data/credential

kube3a k exec secret-pod -- cat /etc/secret-volume/credential
          bob
```

==========

14. Fix a malfunctioning kubernetes cluster.

==========

15. Create a persistent volume with 1GB details provided.
SOLU
====
On k8s.io , seach for "persistentvolume as hostpath" , will see below :
$ pv-1G.yaml
```
kind: PersistentVolume
apiVersion: v1
metadata:
 name: task-pv-volume
 labels:
   type: local
spec:
 storageClassName: manual
 capacity:
   storage: 1Gi
 accessModes:
   - ReadWriteOnce
 hostPath:
   path: "/mnt/data"
```

```
kube3a k create -f pv-1G.yaml
     persistentvolume/task-pv-volume created

kube3a k get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
task-pv-volume   1Gi        RWO            Retain           Available
manual                  2s
kube3a
```

==========

Q 15.1 - Create a pv of size 1GB in namespace qa , using hostpath /srv/boss
SOLU
====

==========

16. Create a pod nginx and expose it.
SOLU
=====

$ k run nginx --image=nginx --restart=Never --port=80 --expose

Above will create nginx pod , with port 80 open in firewall , and will expose as ClusterIP

==========

17. Find all nodes in a cluster which have no taints and store the value into a file.
SOLU
====
kube3a k describe no | egrep -i taint
Taints: node-role.kubernetes.io/master:NoSchedule
Taints: <none>
Taints: <none>
kube3a

$ sudo echo "2" > /opt/XX/k34ft.txt

==========

18. Create a namespace website-backend, run jenkins pod inside it and expose it.
SOLU
=====
$ k create ns website-backend
$ k run jenkins --image=jenkins --restart=Never -n website-jenkins --port=80 --expose

==========

19. ETCD Backup
https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
SOLU
=====

kube3a pwd
     /home/dharmin/etcd-v3.3.12-linux-amd64

```
kube3a ./etcdctl -v
etcdctl version: 3.3.12
API version: 2
kube3a

By default a etcd is in kube-system  namespace
kube3a k get po --all-namespaces | awk '$2 ~ /etcd/ {print $2}'
     etcd-kube3a
kube3a

kube3a k get po etcd-kube3a -n kube-system   --export -o yaml | egrep  "\-file|data-
dir|endpont"
   - --cert-file=/etc/kubernetes/pki/etcd/server.crt          <<== NEED THIS
   - --data-dir=/var/lib/etcd                                  <<== NEED THIS
   - --key-file=/etc/kubernetes/pki/etcd/server.key      <<== NEED THIS
   - --peer-cert-file=/etc/kubernetes/pki/etcd/peer.crt
   - --peer-key-file=/etc/kubernetes/pki/etcd/peer.key
   - --peer-trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt
   - --trusted-ca-file=/etc/kubernetes/pki/etcd/ca.crt   <<== NEED THIS
kube3a

kube3a ./etcdctl backup --help
NAME:
  etcdctl backup - backup an etcd directory

USAGE:
  etcdctl backup [command options]

OPTIONS:
  --data-dir value        Path to the etcd data dir     <<== NEED THIS
  --wal-dir value         Path to the etcd wal dir
  --backup-dir value      Path to the backup dir   <<== NEED THIS
  --backup-wal-dir value  Path to the backup wal dir
  --with-v3               Backup v3 backend data

kube3a

kube3a sudo  ./etcdctl --cert-file /etc/kubernetes/pki/etcd/server.crt --key-file
/etc/kubernetes/pki/etcd/server.key  \
            --ca-file  /etc/kubernetes/pki/etcd/ca.crt  backup --data-dir
/var/lib/etcd --backup-dir /tmp/ABC
[sudo] password for dharmin:
2019-03-20 00:05:36.754230 I | ignoring v3 raft entry
2019-03-20 00:05:36.754479 I | ignoring v3 raft entry
```

```
2019-03-20 00:05:36.754856 I | ignoring v3 raft entry
2019-03-20 00:05:36.755113 I | ignoring v3 raft entry


Lets check out backed-up data :
kube3a sudo chmod -R  755 /tmp/ABC
kube3a

kube3a find /tmp/ABC -print
/tmp/ABC
/tmp/ABC/member
/tmp/ABC/member/wal
/tmp/ABC/member/wal/0000000000000000-0000000000000000.wal
/tmp/ABC/member/snap
/tmp/ABC/member/snap/0000000000000007-00000000000a3973.snap
/tmp/ABC/member/snap/db
kube3a


Lets check if the backed up data is good :
On k8s.io , search for ETCDCTL_API=v3 , and in the pg you see scroll down and shold
see :

kube3a ETCDCTL_API=3 ./etcdctl --write-out=table snapshot status
/tmp/ABC/member/snap/db
+----------+----------+------------+------------+
|   HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| 7fb25e0b |        0 |          1 |      25 kB |
+----------+----------+------------+------------+
kube3a




================================================================================

20. Create nginx as daemonset.
SOLU
=====
Remember a DaemonSet is similar to a Deployment, so create a Deployment manifest
and change is to a daemonset :
kube3a k run redis-ds --image=redis --restart=Always --dry-run -o yaml > redis-
ds.yaml
                                    ^^^^^^^^^^^^^^^^<<== this is optional, don't
```

require

```
$ vi redis-ds.yaml
apiVersion: apps/v1
kind: DaemonSet    <<== this needs to be changed from Deployment to DaemonSet
metadata:
 creationTimestamp: null
 labels:
   run: redis-ds
 name: redis-ds
spec:
 replicas: 1  <<======== Delete this
 selector:
   matchLabels:
     run: redis-ds
 strategy: {}  <<====== Delee this
 template:
   metadata:
     creationTimestamp: null
     labels:
       run: redis-ds
   spec:
     containers:
     - image: redis
       name: redis-ds
       resources: {}  <<== delete this
status: {}   <<<= Delete this

kube3a k create -f  redis-ds.yaml
      daemonset.apps/redis-ds created

kube3a k get ds,po  -o wide
NAME                              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
NODE SELECTOR   AGE   CONTAINERS   IMAGES   SELECTOR
daemonset.extensions/redis-ds   2         2         2       2            2
<none>          35s   redis-ds     redis    run=redis-ds

NAME                 READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE   READINESS GATES
pod/redis-ds-45dd5   1/1     Running   0          35s   10.244.2.152
kube3c.home.local   <none>           <none>
pod/redis-ds-tf5qp   1/1     Running   0          35s   10.244.1.164
kube3b.home.local   <none>           <none>
kube3a
```

==========

21. Scale number of pods of a deployment to 6.
SOLU
=====
$ k run nginx --image=nginx --replicas=2
$ k scale deployment nginx --replicas=6

==========

22. Store the logs of a container having error file-not-found.
SOLU
=====
$ k log <pod_name> | egrep "file-not-found" >   log1
$ sudo cp log1 /opt/XXX/kdrsv

==========

23. Create a pod with local volume details provided.
SOLU
=====
refer to k8s.io and search for hostPath or emptyDir

==========

24. Create a pod and assign it to node labelled as disk=spinning.
SOLU
=====
check if the nodes have the label disk=spinning :
$ k get no ---show-labels

$ k run redis --image=redis --restart=Never --dry-run -o yaml > redis-pod.yaml
then at the end of the line add :
   nodeSelector:
     disk: spinning

Ensure the "nodeSelector:" is in line with "containers:"

==========

if use busybox or alpine pod then always do use --command sleep 3600000 or --
/bin/sh -c 'command'

when comes to nginx or redis pod's , does not require --command or -- /bin/sh   , as per above

PRE-REQ
========

```
$ vi a
alias k=kubectl
alias c=clear

$ source ~/a
$ cp a /tmp
```

When you become root , then :
```
# source /tmp/a
```

0. Create following deployments with 2 pods : nginx, redis, memcached, consul
  expose nginx to service boss
  Forget above question

  Can you tell which pods are part of service boss ?
SOLU
===
If you create a service from a deployment/pod , the default label for service is same as of deployment/pod.

Below will show you Selector , which is same as label :
```
$ kubectl get svc boss --expose -o yaml
```

Once you know what label is assinged to above service , then
```
$ kubectl get po -l <label_name>
```

You'll see the pods that use the service boss.

==========

1. Create nginx static pod , copy the yaml file in /etc/kubernetes/manifest dir.
SOLU
====
On worker Node , as user root :
```
# source /tmp/a
# cd /etc/kubernetes/manifests
```
copy the static pod here from https://kubernetes.io/docs/tasks/administer-cluster/static-pod/

```
# systemctl ststus -l kubelet
kubelet.service - kubelet: The Kubernetes Node Agent
  Loaded: loaded (/lib/systemd/system/kubelet.service; enabled; vendor preset:
enabled)
 Drop-In: /etc/systemd/system/kubelet.service.d
          10-kubeadm.conf
  Active: active (running) since Mon 2019-03-11 11:47:55 GMT; 6s ago
    Docs: https://kubernetes.io/docs/home/
Main PID: 1624 (kubelet)

Our worker Node kube3c is online (Ready) :
kube3a k get no
NAME     STATUS   ROLES    AGE    VERSION
kube3a   Ready    master   5d2h   v1.13.0
kube3b   Ready    <none>   5d1h   v1.13.0
kube3c   Ready    <none>   5d1h   v1.13.0
kube3a
```

==========

14. If you need to know what options to use with securityContext in a pod , how can you find out

SOLU

====

```
     $ kubectl  explain pod.spec.securityContext  --recursive
```

==========

15. Assign a Worker Node with taint tz=tanzania:NoSchedule and assign nginx pod to a node
with the taint tz=tanzania:NoSechedule.

SOLU

====

Show current taint :

```
$ k describe no | egrep -i taints:
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <none>
Taints:             <none>

Add a taint to worker node kube3c :
$ k taint node kube3c  tz=tanzania:NoSchedule
     node/kube3c tainted
```

```
$ k describe no | egrep -i taints:
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <none>
Taints:             tz=tanzania:NoSchedule
$

k run nginx --image=nginx --restart=Never --dry-run -o yaml > nginx-pod1.yaml
$
$ cat nginx-pod1.yaml
apiVersion: v1
kind: Pod
metadata:
 labels:
   run: nginx
 name: nginx
spec:
 containers:
 - image: nginx
   name: nginx
   resources: {}
 dnsPolicy: ClusterFirst
 restartPolicy: Never
 tolerations:             <<== added this
 - key: "tz"              <<== added this
   operator: "Equal"      <<== added this
   value: "tanzania"      <<== added this
   effect: "NoSchedule"   <<== added this
$
$ k apply -f nginx-pod1.yaml
     pod/nginx created

$ k get po -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP               NODE     NOMINATED NODE
READINESS GATES
nginx   1/1     Running   0          43s   192.168.80.221   kube3c   <none>
<none>
$

AS seen above the pod is deployed on kube3c , which has the taint.

Do the clean up Now..
$ k  describe  no kube3c | egrep -i taint
     Taints:             tz=tanzania:NoSchedule
$ k taint node kube3c tz-
```

```
     node/kube3c untainted
$ k  describe  no kube3c | egrep -i taint
     Taints:             <none>
$
Deleted the nginx pod
```

==========

16. Refer to https://kubernetes.io/docs/tasks/debug-application-cluster/core-metrics-pipeline/

PRE-REQ : 1. Install Metric Server on the Master Node

2. Run below commands on Master Node :

```
kubectl run nginx     --image=nginx       --restart=Never -l app=hello
kubectl run redis     --image=redis       --restart=Never -l app=hello
kubectl run consul    --image=consul      --restart=Never -l app=hello
kubectl run memcached --image=memcached   --restart=Never -l app=hello
```

Q. From the pods label app: hello , name the pod that shows high CPU and Memory usage ,
dump the output to /tmp/cpu.usage and /tmp/mem.usage

SOLU

```
$ k top po -l app=hello
NAME        CPU(cores)   MEMORY(bytes)
consul      14m          23Mi
memcached   1m           5Mi
nginx       0m           2Mi
redis       2m           9Mi
$
```

```
$ k top po -l app=hello | sort -nr -k2 | head -1 | awk '{print $1}' | sudo tee /tmp/cpu.usage
                                  ^^<<== refer to 2nd Column

$ k top po -l app=hello | sort -nr -k3 | head -1 | awk '{print $1}' | sudo tee /tmp/mem.usage
```

^^<<== refer to 3rd Column

I am not sure if they want ONLY the pod name or.....
You can :
$ k top po consul | tail -1 | awk '{print $1}' > /tmp/cpu.usage , then edit and only leave 1 pod entry with higher cpu usage
$ k top po consul | tail -1 | awk '{print $1}' > /tmp/cpu.memory , then edit and only leave 1 pod entry with higher Mem usage

==========

17. Sort the PV in ascending order from top to bottom.
SOLU
====

```
$ k get pv  --sort-by=.spec.capacity.storage
NAME    CAPACITY         ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
all-1   2609192632320m   RWX            Retain           Available           manual
65s
all-2   2780991324160m   RWX            Retain           Available           manual
65s
all-3   2791728742400m   RWX            Retain           Available           manual
65s
all-4   3006477107200m   RWX            Retain           Available           manual
65s
all-6   6Gi              RWX            Retain           Available           manual
65s
all-5   6227702579200m   RWX            Retain           Available           manual
65s
$
$
```

17.a Sort all below pv by name

```
     NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
     myvolume         10Gi       RWO,RWX        Retain           Available
normal                  6s
     myvolume1        15Gi       RWO,RWX        Retain           Available
normal                  5s
     pv-1f            100Gi      RWO            Retain           Available
5s
     pv001-1          2Gi        RWO,RWX        Retain           Available
```

```
manual                  6s
     task-pv-volume   1Gi        RWO,RWX        Retain           Available
manual                  6s

SOLU
====
kube3a k get pv --sort-by=.metadata.name
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
myvolume         10Gi       RWO,RWX        Retain           Available
normal                  96s
myvolume1        15Gi       RWO,RWX        Retain           Available
normal                  95s
pv001-1          2Gi        RWO,RWX        Retain           Available
manual                  96s
pv-1f            100Gi      RWO            Retain           Available
95s
task-pv-volume   1Gi        RWO,RWX        Retain           Available
manual                  96s
kube3a
```

==========

18. From etcd pod get the log error msg "store.index" and copy to /opt/KK/xx

```
SOLU
====
      $ k log etcd-kube3a -n kube-system  | egrep "store.index"  | sudo tee
/opt/KK/xx
```

==========

19. Create nginx deployment that has 2 pods, with servicename as nginx-svc , exposed thru nodeport.
   Resolve Pod IP and Service Name in DNS in any container.

```
SOLU
=====
$ k run nginx --image=nginx --replicas=2 --port=80
$ k expose deployment nginx --type=NodePort --port=80 --target-port=80  --
name=nginx-svc
```

After I executed above :

$ k get deploy,svc

```
NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/nginx      2/2     2            2           53s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        6d5h
service/nginx-svc    NodePort    10.103.112.224   <none>        80:30071/TCP   19s
$
```

We know the default label for nginx pod is run=nginx , so lets use the default label and
The best way to get pod IP is :

```
$ k get po -l run=nginx -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
nginx-57867cc648-7n2tc   1/1     Running   0          11m   192.168.79.226   kube3b
<none>           <none>
nginx-57867cc648-bm2vf   1/1     Running   0          11m   192.168.80.222   kube3c
<none>           <none>
$                                                          ^^^^^^^^^^^^^^<<== Pod
IP's
```

Resolve the pod IP's in consule pod :

```
$ k exec consul -- nslookup  192.168.79.226
Name:      192.168.79.226
Address 1: 192.168.79.226 192-168-79-226.nginx-svc.default.svc.cluster.local
$
$ k exec consul -- nslookup  192.168.80.222
Name:      192.168.80.222
Address 1: 192.168.80.222 192-168-80-222.nginx-svc.default.svc.cluster.local
$
```

Resolve the Service Name , the service name is shown when run k get svc :

```
$ k exec consul -- nslookup  nginx-svc
     Name:      nginx-svc
     Address 1: 10.103.112.224 nginx-svc.default.svc.cluster.local
$
```

==========

REF : https://medium.com/@jmarhee/using-initcontainers-to-pre-populate-volume-data-in-kubernetes-99f628cd4519

20. Include busybox initcontainer that creates empty host file in /data , use non persistent volume.

```
apiVersion: v1
kind: Pod
```

```
metadata:
 name: my-app
spec:
 containers:
 - name: my-app
   image: nginx
   volumeMounts:
   - name: config-data
     mountPath: /data
#
SOLU
=====
I created in steps , in first step I made sure the above part is able to create a
pod :
$ cat nginx-pod2.yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-app
spec:
 volumes:
 - name: config-data
   emptyDir: {}
 containers:
 - name: my-app
   image: nginx
   volumeMounts:
   - name: config-data
     mountPath: /data
$

Above pod worked fine..

Next was to create another busybox pod that would create empty hosts file in /data ,
where /data is emptyDir
$ k run bbox --image=busybox --restart=Never --dry-run -o yaml  -- /bin/sh -c
'touch /data/hosts'  > bboxi1.yaml
I extracted info from  bboxi1.yaml and create my final pod manifest.
$ cat  nginx-pod2.yaml
apiVersion: v1
kind: Pod
metadata:
 name: my-app
spec:
 volumes:
```

```
 - name: config-data
   emptyDir: {}
 containers:
 - name: my-app
   image: nginx
   volumeMounts:
   - name: config-data  <<== this matches the name under volume:
     mountPath: /data
# below is from busybox container
 initContainers:
 - args:
   - /bin/sh
   - -c
   - touch /data/hosts
   image: busybox
   name: bbox
   volumeMounts:
   - name: config-data  <<== this matches the name under volume:
     mountPath: /data
# Above is from busybox container
 dnsPolicy: ClusterFirst
 restartPolicy: Never

After I created above pod :
$ k get po
NAME     READY   STATUS    RESTARTS   AGE
my-app   1/1     Running   0          10m

$ k exec my-app --  ls /data/hosts
     /data/hosts

$ k exec my-app --  ls  -l /data/hosts
     -rw-r--r-- 1 root root 0 Mar 13 14:30 /data/hosts
$
Time to delete my pod :
$ k delete po my-app
     pod "my-app" deleted
$

================================================================================

REF : https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ic/
deploy.yaml

21 . Create alpine init container so can be used with deploy , do create path to
```

```
local disk.
apiVersion: apps/v1beta1
kind: Deployment
metadata:
 name: ic-deploy
spec:
 replicas: 1
 template:
   metadata:
     labels:
       app: ic
   spec:
     containers:
     - name: main
       image: centos:7
       command:
       - "bin/bash"
       - "-c"
       - "while true; do cat /ic/this; sleep 5; done"
       volumeMounts:
       - mountPath: /ic
         name: msg

SOLU
=====
Loking at above manifest, seems like volume is missing , so lets add that  , and
run :  k run alpine --image=alpine --restart=Never --dry-run -o yaml  -- /bin/sh -c
'echo BOSSSS > /ic/this' > alpine.yaml
use info from alpine.yaml file and add in below deployment.

$ cat  ic-deploy.yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
 name: ic-deploy
spec:
 replicas: 1
 template:
   metadata:
     labels:
       app: ic
   spec:
     volumes:
     - name: msg
       hostPath:
```

```
        path: /mnt/data
    containers:
    - name: main
      image: centos:7
      command:
      - "bin/bash"
      - "-c"
      - "while true; do cat /ic/this; sleep 5; done"
      volumeMounts:
      - mountPath: /ic
        name: msg
    initContainers:
    - args:
      - /bin/sh
      - -c
      - echo BOSSS > /ic/this
      image: alpine
      name: alpine
      volumeMounts:
      - mountPath: /ic
        name: msg
##      dnsPolicy: ClusterFirst  <<== this is part of the alpine pod,
##      restartPolicy: Never     <<== was generated when created alpine pod
$
$ k get deploy,po
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/ic-deploy      1/1     1            1           32m

NAME                                 READY   STATUS    RESTARTS   AGE
pod/ic-deploy-786f5d879d-pckxg       1/1     Running   0          32m
$

$ k exec ic-deploy-786f5d879d-pckxg -- cat /ic/this
    BOSSS
$
$ k logs ic-deploy-786f5d879d-pckxg
    BOSSS


================================================================================

22.  Add initContainer and non persistent disk
apiVersion: v1
kind: Pod
metadata:
```

```
 name: happypanda
spec:
 containers:
 - name: busybox
   image: busybox:latest
   command: ["/bin/sh", "-c"]
   args: ["cat /opt/workdir/helloworld && sleep 3600"]
   volumeMounts:
   - name: workdir
     mountPath: /opt/workdir

SOLU
=====

$ k run alpine  --image=alpine  --restart=Never  --dry-run  -o yaml -- /bin/sh -c
'echo BOSSS > /opt/workdir/helloworld'

$ cat initcon.yaml
apiVersion: v1
kind: Pod
metadata:
 name: happypanda
spec:
 volumes:
 - name: workdir
   emptyDir: {}   <<== this is non perisstent disk
#
 containers:
 - name: busybox
   image: busybox
   command: ["/bin/sh", "-c"]
   args: ["cat /opt/workdir/helloworld && sleep 3600"]
   volumeMounts:
   - name: workdir
     mountPath: /opt/workdir
#
# Below was generated from above "k run alpine ........"
 initContainers:
 - args:
   - /bin/sh
   - -c
   - echo BOSSS > /opt/workdir/helloworld
   image: alpine
   name: alpine
   volumeMounts:               <<== This I copied from above
```

```
   - name: workdir             <<==  """""""""""""""""""""""
     mountPath: /opt/workdir   <<== """""""""""""""""""""""
##
##  dnsPolicy: ClusterFirst  <<== this is part of the alpine pod , don't require
##  restartPolicy: Never     <<==  """""""""""""""""""""""""""""""""""""""""""""
$

After I deployed above pod , then :
$ k get po  happypanda
NAME         READY   STATUS    RESTARTS   AGE
happypanda   1/1     Running   0          4m17s

I ran "k describe po happypanda"    , and only showing you few lines..
Events:
 Type    Reason     Age    From               Message
 ----    ------     ----   ----               -------
 Normal  Scheduled  5m14s  default-scheduler  Successfully assigned
default/happypanda to kube3c
 Normal  Pulling    5m13s  kubelet, kube3c    pulling image "alpine"
 Normal  Pulled     5m6s   kubelet, kube3c    Successfully pulled image "alpine"
 Normal  Created    5m6s   kubelet, kube3c    Created container
 Normal  Started    5m6s   kubelet, kube3c    Started container
 Normal  Pulling    5m6s   kubelet, kube3c    pulling image "busybox"
 Normal  Pulled     5m4s   kubelet, kube3c    Successfully pulled image "busybox"
 Normal  Created    5m4s   kubelet, kube3c    Created container
 Normal  Started    5m4s   kubelet, kube3c    Started container
$

================================================================================

23.  List a Node with high CPU Usage and copy the output to /opt/TTT/cpu.higg.usage
    List a Node with High Memory Usage and save the output to
/opt/TTT/mem.usage.high

SOLU
=====
BECOME ROOT , then execute the kubectl config user-config <cluster_name>   , then :
if kubectl command does not work then may need to :
KUBECONFIG=/etc/kubernetes/admin.conf

root@kube3a:~# kubectl top no
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
kube3a              73m          3%     1167Mi          14%
kube3b.home.local   15m          0%     326Mi           4%
kube3c.home.local   16m          0%     355Mi           4%
```

```
root@kube3a:~#
root@kube3a:~# kubectl top no | sort -rn -k2 | head -1 | awk '{print $1}'
     kube3a                            ^^<<== -k is the column no.

root@kube3a:~# kubectl top no | sort -rn -k2 | head -1 | awk '{print $1}'  >
/opt/TTT/cpu.higg.usage
root@kube3a:~# cat /opt/TTT/cpu.higg.usage
          kube3a
root@kube3a:~#


root@kube3a:~# kubectl top no | sort -rn -k4 | head -1 | awk '{print $1}'
          kube3a
root@kube3a:~# kubectl top no | sort -rn -k4 | head -1 | awk '{print $1}' >
/opt/TTT/mem.usage.high

root@kube3a:~# cat /opt/TTT/mem.usage.high
          kube3a


================================================================================

24.  Count number of Nodes that are in ready state , except  ones which shows
tolerance : noschedule
    save the results in /opt/TTT/node-ready
SOLU
====
As we don't have perm to wrie in /opt/TTT dir , for that we'll need to become
root :
$ cp a /tmp/a
kube3a sudo -i
root@kube3a:~#
root@kube3a:~# source /tmp/a
kube3a
kube3a id -a
     uid=0(root) gid=0(root) groups=0(root)
kube3a

copy & paste from top of the Q : kubectl config use-config <cluster_name>

kube3a k get no
The connection to the server localhost:8080 was refused - did you specify the right
host or port?
kube3a export  KUBECONFIG=/etc/kubernetes/admin.conf
```

```
kube3a

kube3a k get no
NAME                STATUS   ROLES    AGE   VERSION
kube3a              Ready    master   14d   v1.13.1
kube3b.home.local   Ready    <none>   14d   v1.13.1
kube3c.home.local   Ready    <none>   9d    v1.13.1

kube3a k describe no | egrep -i taint:
kube3a k describe no | egrep -i taint
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <none>
Taints:             <none>
kube3a

kube3a echo  2 > /opt/TTT/node-ready
kube3a cat /opt/TTT/node-ready
     2
kube3a
```

==========

25. You are unable to Query the cluster nodes :

```
     root@kube3a:~# kubectl get no
          The connection to the server localhost:8080 was refused - did you specify
the right host or port?
```

SOLU

=====

If kubelet service is down you can also get above error. Run :

```
     systemctl status -k kubelet
```

You need to check 2 files , as the /etc/{BROKEN|broken] could be in either of the file.

```
$ sudo  vi  /etc/systemd/system/kubelet.service
```

OR

```
$ sudo  vi  /lib/systemd/system/kubelet.service
```

REPLACE : /etc/[BROKEN|broken] OR /etc/kuernetes/BROEKN

WITH : /etc/kubernetes/manifests

```
$ sudo systemctl daemon-reload
$ systemctl status -l kubelet
```

OR

```
$ systemctl restart kubelet
```

ALso check that the kubelete service is enabled :
$ systemctl list-unit-files | egrep -i kubelet

Here you'll need to wait for say 2mins , before able to run command kubectl get no

==========

26. Create a deployment which contains one nginx container. The deployment needs a label
  called app:frontend and should be configured for 5 replicas.
 Expose a deployment with clusterIP service that listens on tcp/8080 and target tcp/8080.
SOLU
====

==========

27.
1. create a pod named "web" using image nginx:1.11.9-alpine, on ports 80 and 443
SOLU
=====
kube3a k run web --image=nginx:1.11.9-alpine --restart=Never --port=80 --dry-run -o yaml > nG.yaml
kube3a

kube3a vi nG.yaml
apiVersion: v1
kind: Pod
metadata:
 creationTimestamp: null
 labels:
   run: web
 name: web
spec:
 containers:
 - image: nginx:1.11.9-alpine
   name: web
   ports:
   - containerPort: 80
   - containerPort: 443   <<== I manually added this entry
   resources: {}
 dnsPolicy: ClusterFirst

```
 restartPolicy: Never
status: {}

kube3a k create -f nG.yaml
     pod/web created

kube3a k get po
NAME   READY   STATUS    RESTARTS   AGE
web    1/1     Running   0          15s
kube3a


kube3a k describe po web | egrep -i port
   Ports:          80/TCP, 443/TCP
   Host Ports:     0/TCP, 0/TCP
kube3a


2. create a service to expose that pod, named as "webservice"
SOLU
====
kube3a k expose  pod web --port=80,443 --name=webservice
     service/webservice exposed

kube3a k get svc  webservice
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)           AGE
webservice   ClusterIP   10.107.250.6   <none>        80/TCP,443/TCP   8s
kube3a


3. copy the dns records for the service in file /opt/TTT/web.dnsrecord
  record the pod dns in /opt/TTT/pod.dns
SOLU
====
kube3a k run bbox --image=busybox:1.28 --restart=Never  -- /bin/sh -c 'sleep 36000'
     pod/bbox created

kube3a k get po -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP            NODE
NOMINATED NODE   READINESS GATES
bbox   1/1     Running   0          45s   10.244.1.10   kube3b.home.local   <none>
<none>
web    1/1     Running   0          11m   10.244.2.25   kube3c.home.local   <none>
<none>
kube3a                                  ^^^^^^^^^^<<== this is the pod IP , which
```

needs to resolve as pod dns

```
kube3a k exec bbox -- nslookup 10.244.2.25 | sudo tee /opt/TTT/pod.dns
[sudo] password for dharmin:
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.244.2.25
Address 1: 10.244.2.25 10-244-2-25.webservice.default.svc.cluster.local
kube3a


kube3a k get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP          21d
webservice   ClusterIP   10.107.250.6   <none>        80/TCP,443/TCP   12m
kube3a

Here we resolving DNS name of service webservice :
kube3a k exec  bbox -- nslookup webservice | sudo tee /opt/TTT/web.dnsrecord
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      webservice
Address 1: 10.107.250.6 webservice.default.svc.cluster.local
kube3a
```

==========

One i found which wasn't there was
Finding all nodes which are in ready status (except those where tolerance : noschedule)
Also - they answer that Sanjay told to restore the faulty node also worked
Sol - open kubelet systemd file n you will see /etc/ BROKEN will be written
Just remove broken n add /Kubernetes/manifests
Do daemon reload n kubelet restart

Note - this will take time
My advice to u will be do write ASAP
All questions are same
I'm sure you will pass
One more thing - we have to only find out ONE node that has HIGH cpu utilization
Not all the nodes

==========

Kubectl get node OR pod - you'll see Connection Refused Error.

One Master and One Worker Node.

Login to the FIRST Node

Open the Kubelet Service File.

sudo systemctl status kubelet

Open this file : /lib/systemd/system/kubelet.service

Similar to the

-pod-manifest-path

In that file, you'll see

/etc/broken

You've to change is this to /etc/kubernetes/manifest

==========

NOTE :
For metric server you have to include below :

```
- command:
  - /metrics-server
  - --kubelet-insecure-tls
  - --kubelet-preferred-address-types=InternalIP
  - --logtostderr
```