

# Linux-Foundation

## Exam Questions CKA

Certified Kubernetes Administrator (CKA) Program



### NEW QUESTION 1

Create a deployment as follows:

- > Name:nginx-random
- > Exposed via a service nginx-random
- > Ensure that the service & pod are accessible via their respective DNS records
- > The container(s) within any pod(s) running as a part of this deployment should use the nginx image

Next, use the utility `nslookup` to lookup the DNS records of the service & pod and write the output to `/opt/KUNW00601/service.dns` and `/opt/KUNW00601/pod.dns` respectively.

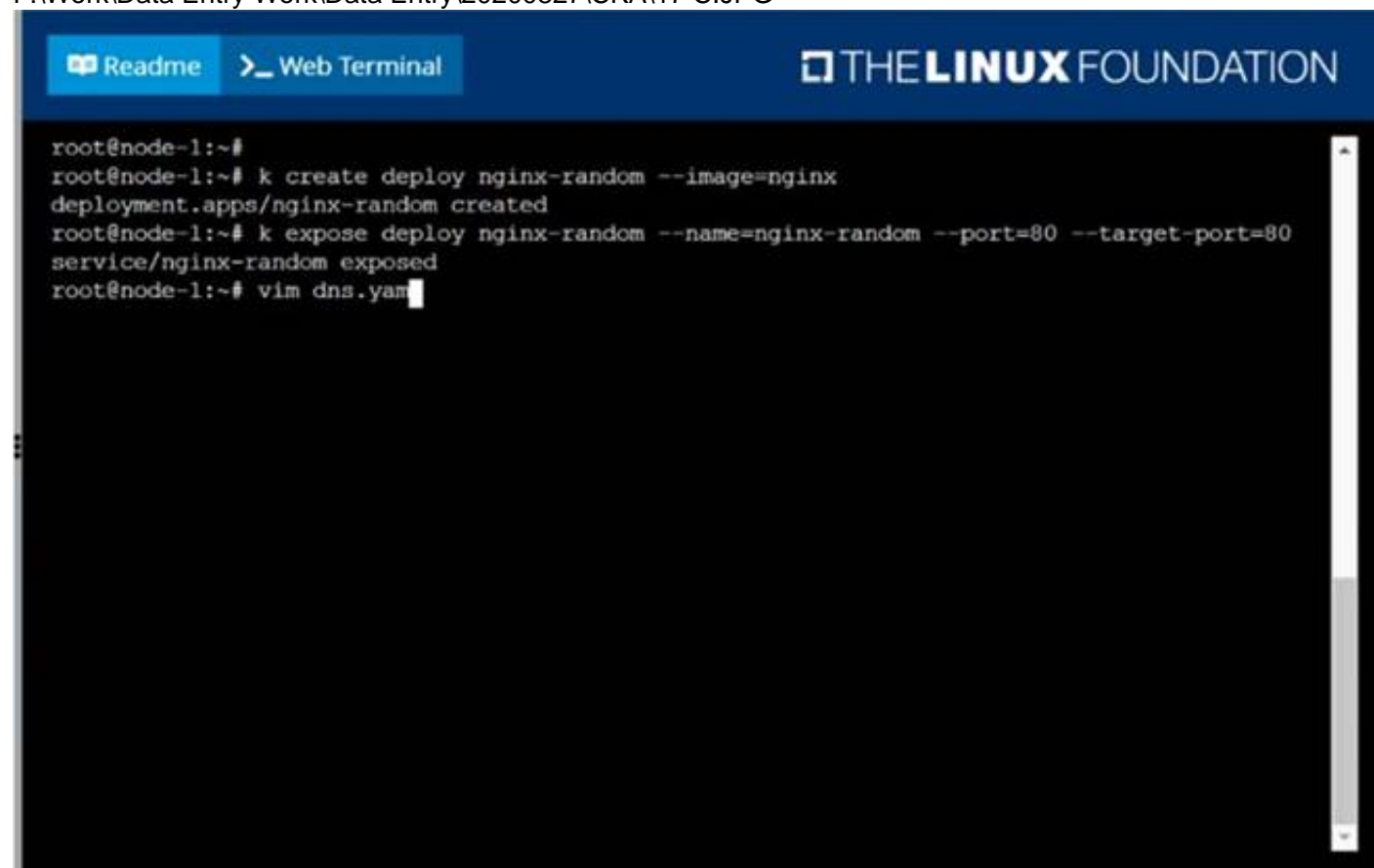
- A. Mastered
- B. Not Mastered

**Answer: A**

**Explanation:**

Solution:

F:\Work\Data Entry Work\Data Entry\20200827\CKA\17 C.JPG



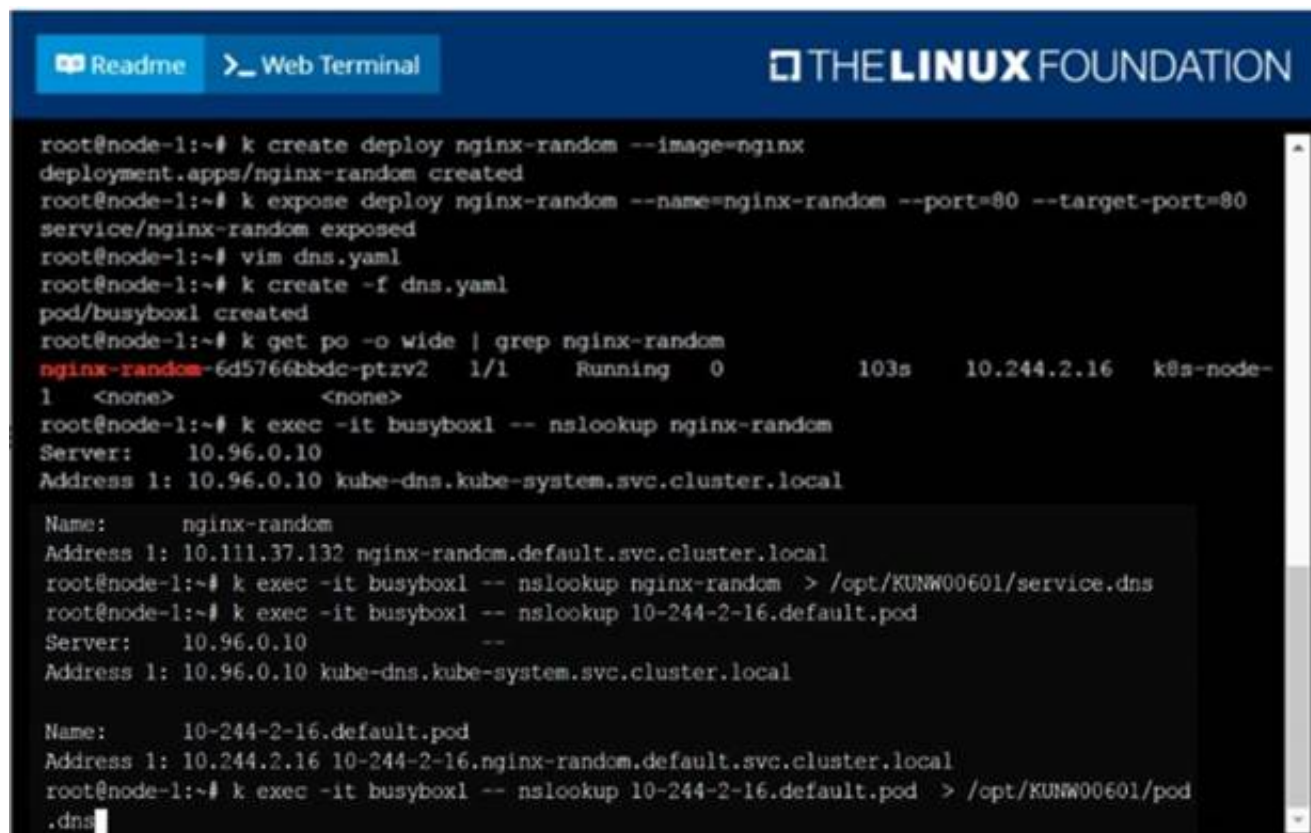
```
root@node-1:~#
root@node-1:~# k create deploy nginx-random --image=nginx
deployment.apps/nginx-random created
root@node-1:~# k expose deploy nginx-random --name=nginx-random --port=80 --target-port=80
service/nginx-random exposed
root@node-1:~# vim dns.yaml
```

F:\Work\Data Entry Work\Data Entry\20200827\CKA\17 D.JPG



```
apiVersion: v1
kind: Pod
metadata:
  name: busybox1
  labels:
    name: busybox
spec:
  containers:
  - image: busybox:1.28
    command:
      - sleep
      - "3600"
    name: busybox
```

F:\Work\Data Entry Work\Data Entry\20200827\CKA\17 E.JPG



```
root@node-1:~# k create deploy nginx-random --image=nginx
deployment.apps/nginx-random created
root@node-1:~# k expose deploy nginx-random --name=nginx-random --port=80 --target-port=80
service/nginx-random exposed
root@node-1:~# vim dns.yaml
root@node-1:~# k create -f dns.yaml
pod/busybox1 created
root@node-1:~# k get po -o wide | grep nginx-random
nginx-random-6d5766bbdc-ptzv2 1/1 Running 0 103s 10.244.2.16 k8s-node-1 <none> <none>
root@node-1:~# k exec -it busybox1 -- nslookup nginx-random
Server: 10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: nginx-random
Address 1: 10.111.37.132 nginx-random.default.svc.cluster.local
root@node-1:~# k exec -it busybox1 -- nslookup nginx-random > /opt/KUNW00601/service.dns
root@node-1:~# k exec -it busybox1 -- nslookup 10-244-2-16.default.pod
Server: 10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: 10-244-2-16.default.pod
Address 1: 10.244.2.16 10-244-2-16.nginx-random.default.svc.cluster.local
root@node-1:~# k exec -it busybox1 -- nslookup 10-244-2-16.default.pod > /opt/KUNW00601/pod.dns
```

## NEW QUESTION 2

List the nginx pod with custom columns POD\_NAME and POD\_STATUS

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

kubect! get po -o=custom-columns="POD\_NAME:.metadata.name, POD\_STATUS:.status.containerStatuses[.state"

## NEW QUESTION 3

Create a pod namedkucc8with asingle app container for each of the following images running inside(there may be between 1 and 4images specified): nginx + redis + memcached.

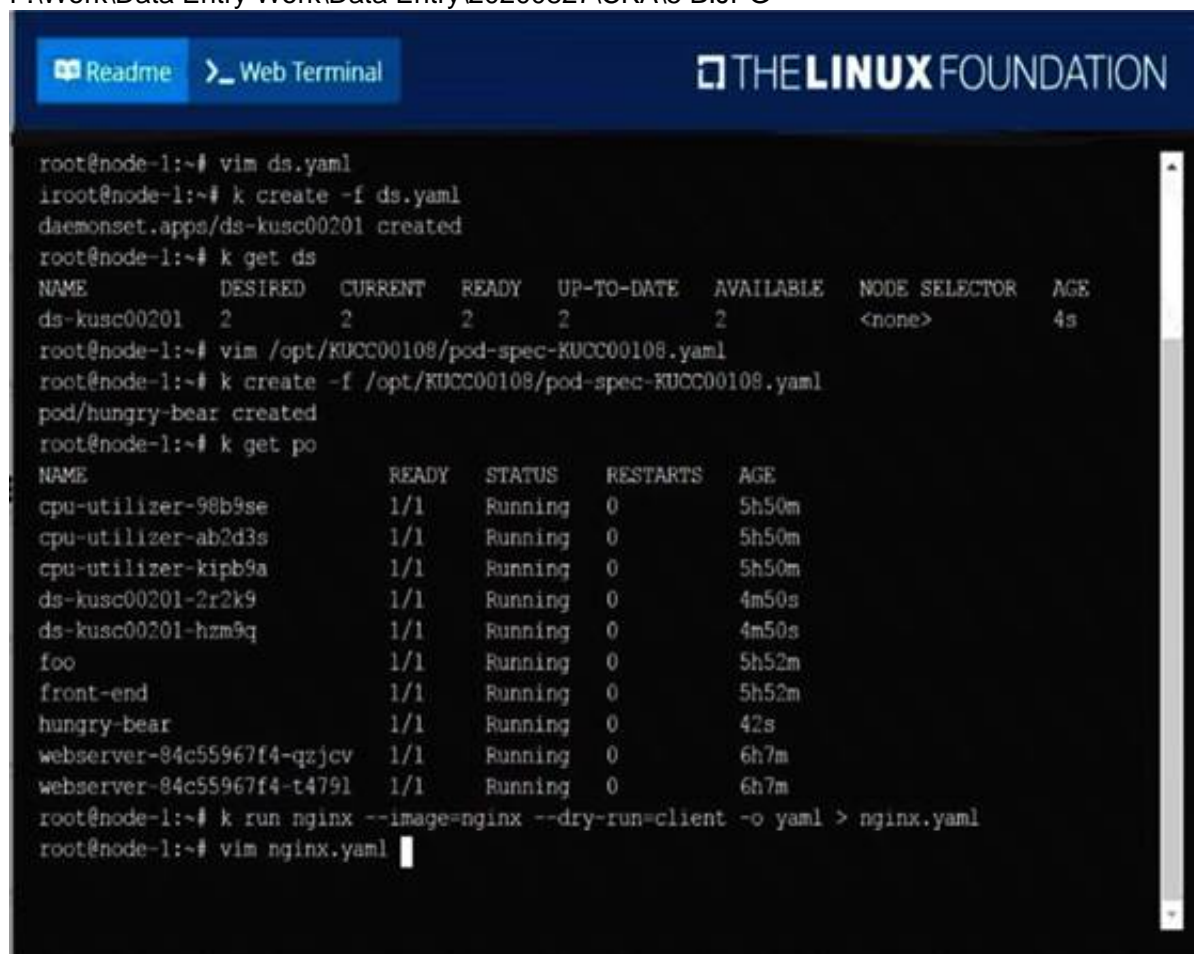
- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

solution

F:\Work\Data Entry Work\Data Entry\20200827\CKA\5 B.JPG

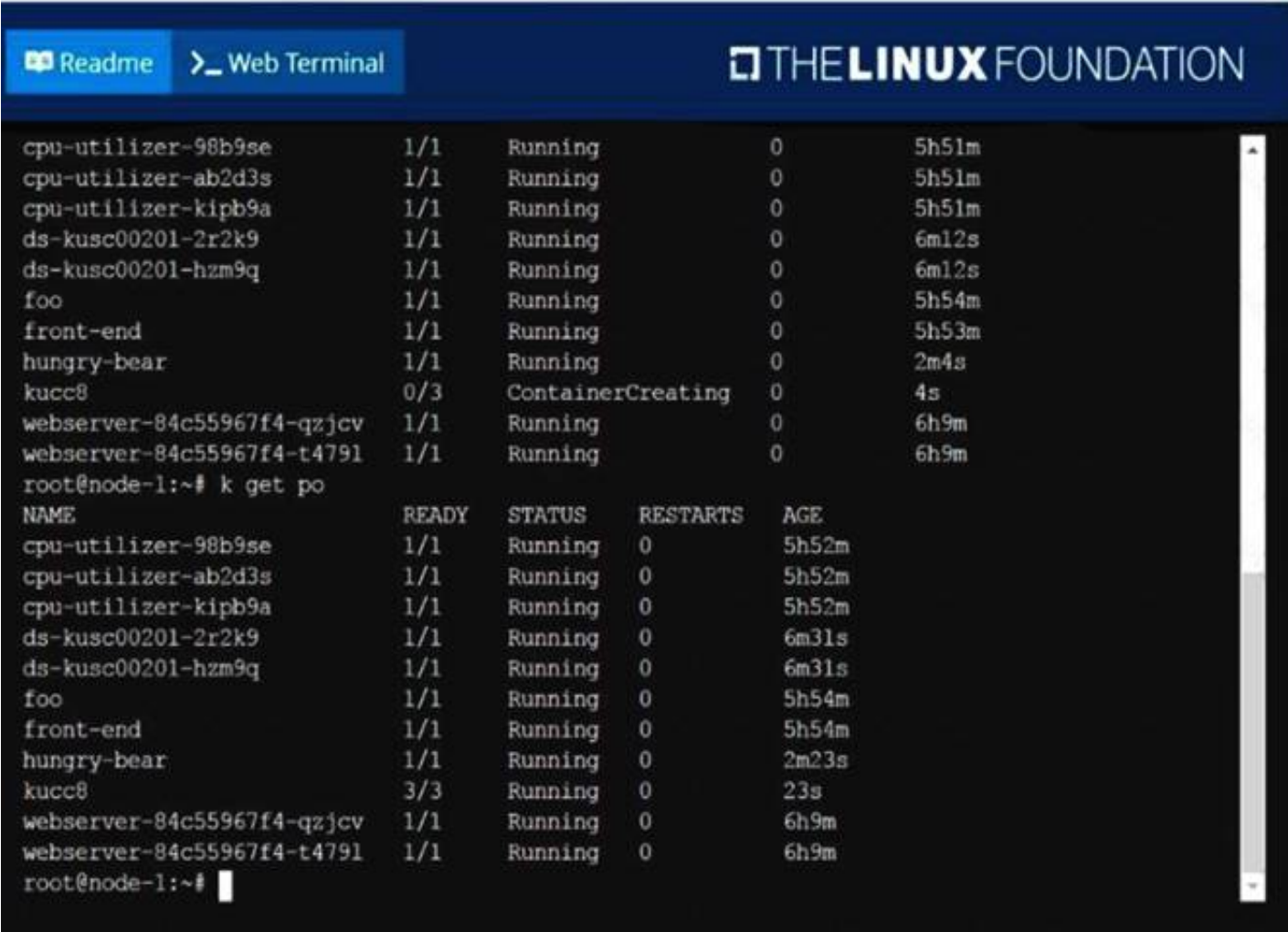


```
root@node-1:~# vim ds.yaml
root@node-1:~# k create -f ds.yaml
daemonset.apps/ds-kusc00201 created
root@node-1:~# k get ds
NAME          DESIRED  CURRENT  READY  UP-TO-DATE  AVAILABLE  NODE SELECTOR  AGE
ds-kusc00201  2        2        2      2          2          <none>         4s
root@node-1:~# vim /opt/KUCC00108/pod-spec-KUCC00108.yaml
root@node-1:~# k create -f /opt/KUCC00108/pod-spec-KUCC00108.yaml
pod/hungry-bear created
root@node-1:~# k get po
NAME          READY  STATUS   RESTARTS  AGE
cpu-utilizer-98b9se  1/1    Running  0         5h50m
cpu-utilizer-ab2d3s  1/1    Running  0         5h50m
cpu-utilizer-kipb9a  1/1    Running  0         5h50m
ds-kusc00201-2r2k9  1/1    Running  0         4m50s
ds-kusc00201-hzm9q  1/1    Running  0         4m50s
foo            1/1    Running  0         5h52m
front-end       1/1    Running  0         5h52m
hungry-bear     1/1    Running  0         42s
webserver-84c55967f4-qzjcv  1/1    Running  0         6h7m
webserver-84c55967f4-t479l  1/1    Running  0         6h7m
root@node-1:~# k run nginx --image=nginx --dry-run=client -o yaml > nginx.yaml
root@node-1:~# vim nginx.yaml
```

F:\Work\Data Entry Work\Data Entry\20200827\CKA\5 C.JPG



F:\Work\Data Entry Work\Data Entry\20200827\CKA\5 D.JPG



#### NEW QUESTION 4

Create a pod with image nginx called nginx and allow traffic on port 80

- A. Mastered
- B. Not Mastered

Answer: A

#### Explanation:

kubectlrn nginx --image=nginx --restart=Never --port=80

#### NEW QUESTION 5

Create a nginx pod with label env=test in engineering namespace

- A. Mastered
- B. Not Mastered



**Answer:** A

**Explanation:**

```
kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml > nginx-pod.yaml
kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml | kubectl create -nengineering -f ?C
YAML File: apiVersion: v1 kind: Pod metadata: name: nginx
namespace: engineering labels:
env: test spec: containers:
- name: nginx image: nginx
imagePullPolicy: IfNotPresent restartPolicy: Never
kubectl create -f nginx-pod.yaml
```

**NEW QUESTION 6**

Create a persistent volume with name app-data, of capacity 2Gi and access mode ReadWriteMany. The type of volume is hostPath and its location is /srv/app-data.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

solution  
 Persistent Volume  
 A persistent volume is a piece of storage in a Kubernetes cluster. PersistentVolumes are a cluster-level resource like nodes, which don't belong to any namespace. It is provisioned by the administrator and has a particular file size. This way, a developer deploying their app on Kubernetes need not know the underlying infrastructure. When the developer needs a certain amount of persistent storage for their application, the system administrator configures the cluster so that they consume the PersistentVolume provisioned in an easy way.  
 Creating PersistentVolume  
 kind: PersistentVolume apiVersion: v1 metadata: name: app-data spec: capacity: # defines the capacity of PV we are creating storage: 2Gi # the amount of storage we are trying to claim accessModes: # defines the rights of the volume we are creating - ReadWriteMany hostPath: path: "/srv/app-data" # path to which we are creating the volume  
 Challenge  
 > Create a Persistent Volume named app-data, with access mode ReadWriteMany, storage class name shared, 2Gi of storage capacity and the host path /srv/app-data.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /srv/app-data
  storageClassName: shared
```

"app-data.yaml" 12L, 194C

\* 2. Save the file and create the persistent volume. Image for post

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl create -f pv.yaml
persistentvolume/pv created
```

\* 3. View the persistent volume.

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
app-data	2Gi	RWX	Retain	Available		shared		31s

> Our persistent volume status is available meaning it is available and it has not been mounted yet. This status will change when we mount the persistentVolume to a persistentVolumeClaim.

PersistentVolumeClaim

In a real ecosystem, a system admin will create the PersistentVolume then a developer will create a PersistentVolumeClaim which will be referenced in a pod. A PersistentVolumeClaim is created by specifying the minimum size and the access mode they require from the persistentVolume.

Challenge

> Create a Persistent Volume Claim that requests the Persistent Volume we had created above. The claim should request 2Gi. Ensure that the Persistent Volume Claim has the same storageClassName as the persistentVolume you had previously created.  
kind: PersistentVolumeapiVersion: v1metadata:name: app-data spec:  
accessModes: -ReadWriteManyresources:  
requests: storage: 2Gi storageClassName: shared  
\* 2. Save and create the pvc  
njerry191@cloudshell:~(extreme-clone-265411)\$ kubectl create -f app-data.yaml persistentvolumeclaim/app-data created  
\* 3. View the pvc Image for post

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl get pvc
NAME      STATUS    VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
pv        Bound     pv           512m       RWX             shared
```

\* 4. Let's see what has changed in the pv we had initially created.  
Image for post

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM      STORAGECLASS   REASON   AGE
pv        512m       RWX            Retain            Bound    default/pv  shared         16m
```

Our status has now changed from available to bound.

\* 5. Create a new pod named myapp with image nginx that will be used to Mount the Persistent Volume Claim with the path /var/app/config.  
Mounting a Claim  
apiVersion: v1kind: Podmetadata: creationTimestamp: nullname: app-dataspec: volumes: - name: configpvcpersistentVolumeClaim: claimName: app-datacontainers: -  
image: nginxname: appvolumeMounts: - mountPath: "/srv/app-data"name: configpvc

NEW QUESTION 7

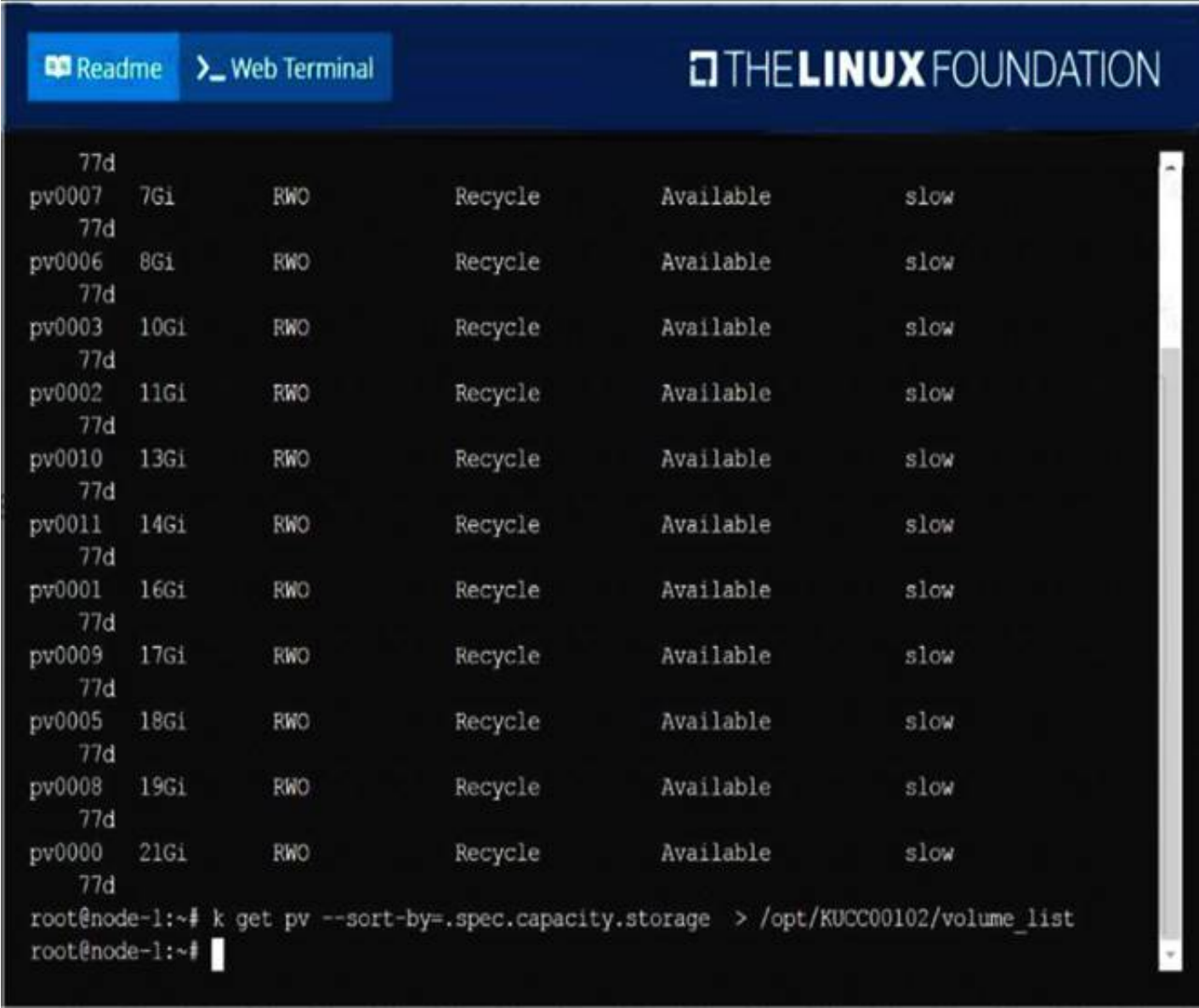
List all persistent volumes sorted by capacity, saving the full kubectl output to /opt/KUCC00102/volume\_list. Use kubectl's own functionality for sorting the output, and do not manipulate it any further.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

solution  
F:\Work\Data Entry Work\Data Entry\20200827\CKA\2 C.JPG



NEW QUESTION 8

A Kubernetes worker node, named dwk8s-node-0 is in state NotReady. Investigate why this is the case, and perform any appropriate steps to bring the node to a Ready state, ensuring that any changes are made permanent.  
You can ssh to the failed node using:

[student@node-1] \$ | sshWk8s-node-0

You can assume elevated privileges on the node with the following command:

[student@wk8s-node-0] \$ | sudo ?Ci

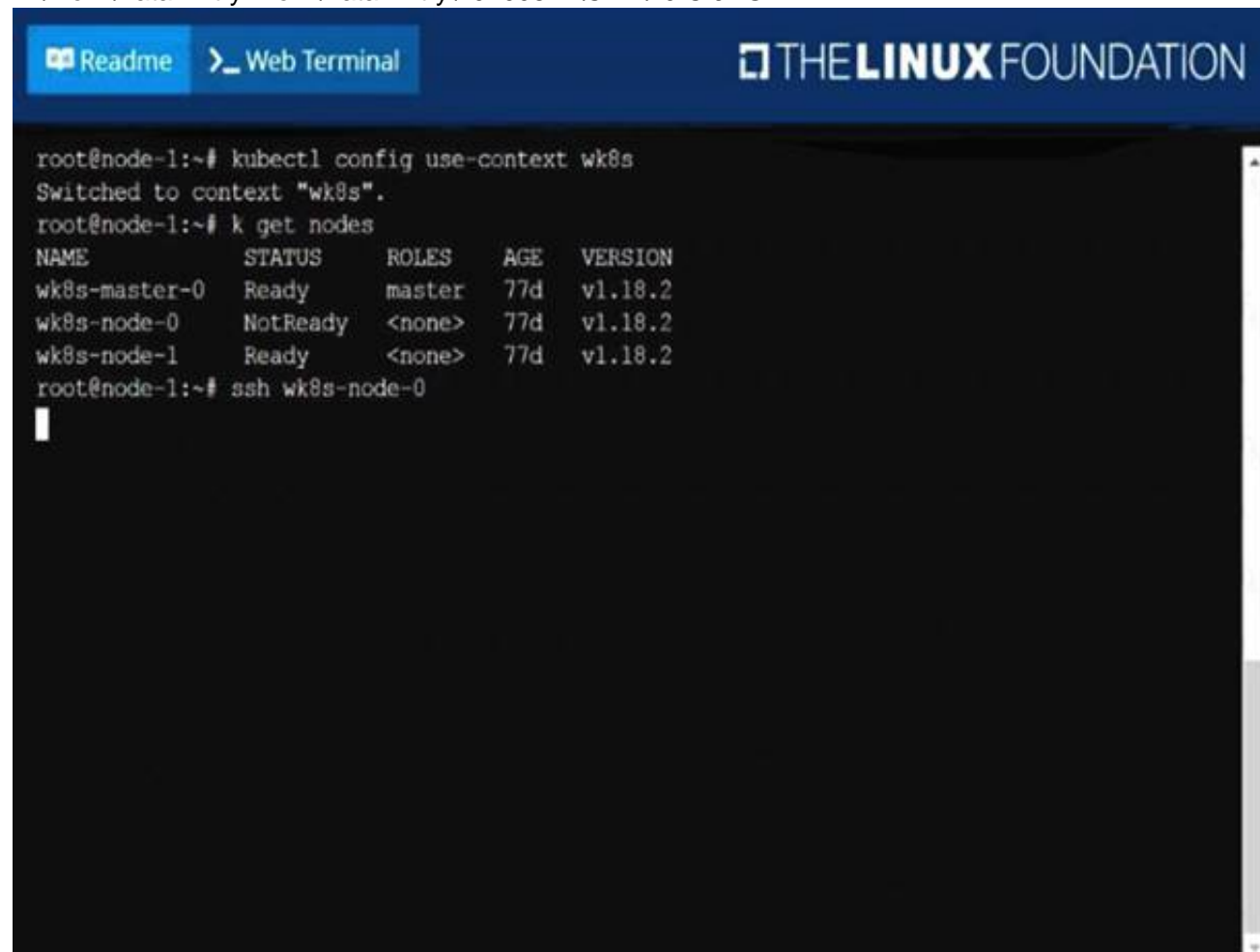
- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

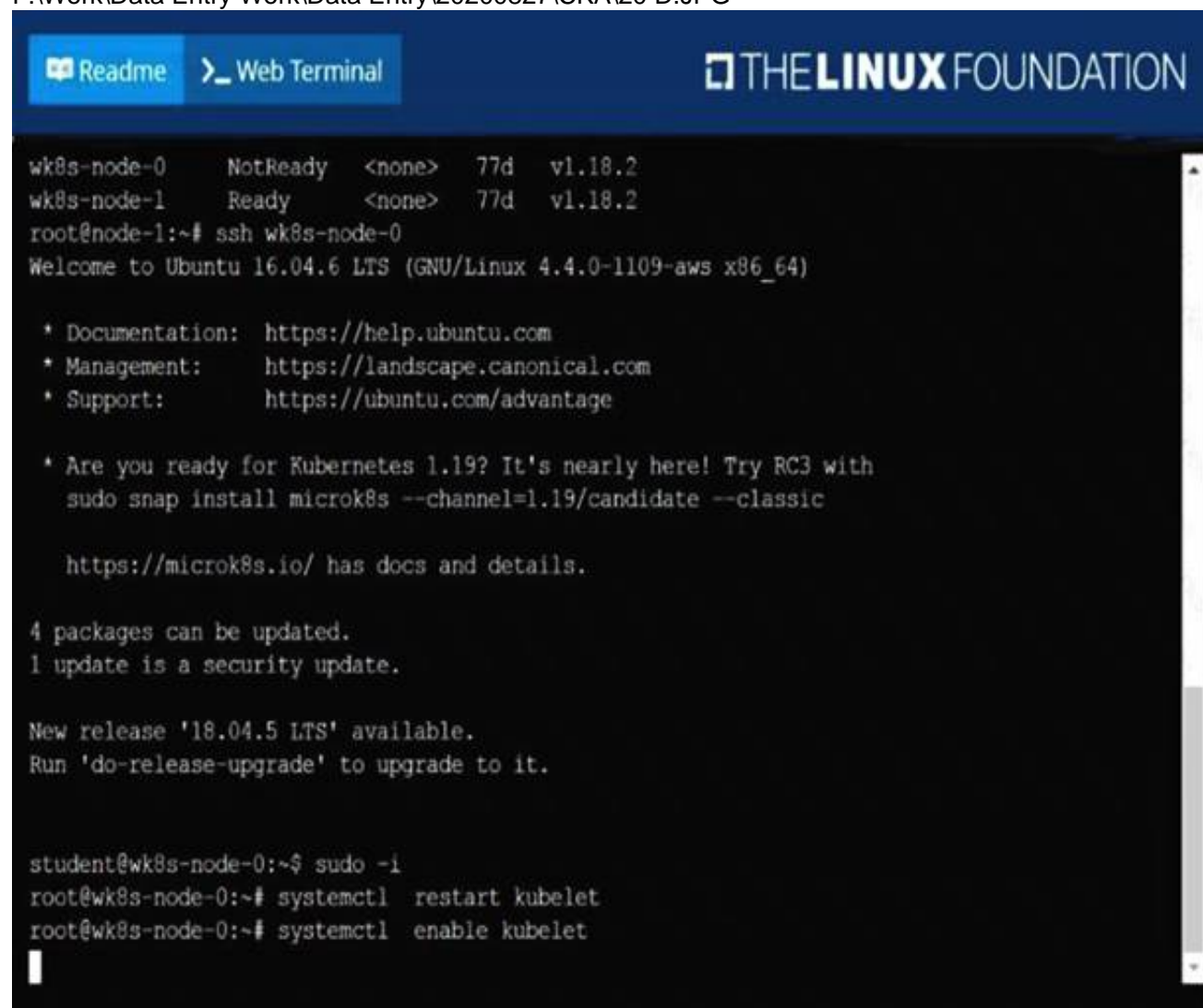
solution

F:\Work\Data Entry Work\Data Entry\20200827\CKA\20 C.JPG



```
root@node-1:~# kubectl config use-context wk8s
Switched to context "wk8s".
root@node-1:~# k get nodes
NAME             STATUS    ROLES    AGE   VERSION
wk8s-master-0    Ready     master   77d   v1.18.2
wk8s-node-0      NotReady  <none>    77d   v1.18.2
wk8s-node-1      Ready     <none>    77d   v1.18.2
root@node-1:~# ssh wk8s-node-0
█
```

F:\Work\Data Entry Work\Data Entry\20200827\CKA\20 D.JPG



```
wk8s-node-0    NotReady  <none>    77d   v1.18.2
wk8s-node-1    Ready     <none>    77d   v1.18.2
root@node-1:~# ssh wk8s-node-0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic
   https://microk8s.io/ has docs and details.


4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-0:~$ sudo -i
root@wk8s-node-0:~# systemctl restart kubelet
root@wk8s-node-0:~# systemctl enable kubelet
█
```

F:\Work\Data Entry Work\Data Entry\20200827\CKA\20 E.JPG





```

https://microk8s.io/ has docs and details.

4 packages can be updated.
1 update is a security update.

New release '18.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

student@wk8s-node-0:~$ sudo -i
root@wk8s-node-0:~# systemctl restart kubelet
root@wk8s-node-0:~# systemctl enable kubelet
Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /lib/systemd/system/kubelet.service.
root@wk8s-node-0:~# exit
logout
student@wk8s-node-0:~$ exit
logout
Connection to 10.250.5.34 closed.
root@node-1:~# k get nodes
NAME             STATUS    ROLES    AGE   VERSION
wk8s-master-0    Ready     master   77d   v1.18.2
wk8s-node-0      Ready     <none>   77d   v1.18.2
wk8s-node-1      Ready     <none>   77d   v1.18.2
root@node-1:~#

```

#### NEW QUESTION 9

Get list of all pods in all namespaces and write it to file ??/opt/pods-list.yaml??

- A. Mastered
- B. Not Mastered

**Answer:** A

#### Explanation:

kubectkl get po ?Call-namespaces > /opt/pods-list.yaml

#### NEW QUESTION 10

Check the image version in pod without the describe command

- A. Mastered
- B. Not Mastered

**Answer:** A

#### Explanation:

kubectkl get po nginx -o jsonpath='{.spec.containers[].image}{"\n"}'

#### NEW QUESTION 10

Create 2 nginx image pods in which one of them is labelled with env=prod and another one labelled with env=dev and verify the same.

- A. Mastered
- B. Not Mastered

**Answer:** A

#### Explanation:

kubectkl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like ??creationTimestamp: null?? ??dnsPolicy: ClusterFirst??

vim nginx-prod-pod.yaml apiVersion: v1

kind: Pod metadata: labels: env: prod

name: nginx-prod spec:

containers:

- image: nginx name: nginx-prod

restartPolicy: Always

# kubectkl create -f nginx-prod-pod.yaml

kubectkl run --generator=run-pod/v1 --image=nginx -- labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml apiVersion: v1

kind: Pod metadata: labels: env: dev

name: nginx-dev

spec: containers:

- image: nginx name: nginx-dev

restartPolicy: Always

# kubectkl create -f nginx-prod-dev.yaml Verify :



```
kubectl get po --show-labels kubectl get po -l env=prod kubectl get po -l env=dev
```

**NEW QUESTION 14**

Create a busybox pod and add `sleep 3600` command

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

```
kubectl run busybox --image=busybox --restart=Never -- /bin/sh -c "sleep 3600"
```

**NEW QUESTION 18**

List all the pods sorted by name

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

```
kubectl get pods --sort-by=.metadata.name
```

**NEW QUESTION 21**

For this item, you will have to ssh to the `nodeik8s-master-0` and `nodeik8s-node-0` and complete all tasks on these nodes. Ensure that you return to the base node (`hostname=node-1`) when you have completed this item.

Context

As an administrator of a small development team, you have been asked to set up a Kubernetes cluster to test the viability of a new application.

Task

You must use `kubeadm` to perform this task. Any `kubeadm` invocations will require the use of the `--ignore-preflight-errors=alloption`.

- > Configure the `nodeik8s-master-0` as a master node. .
- > Join the `nodeik8s-node-0` to the cluster.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

solution

You must use the `kubeadm` configuration file located at `/etc/kubeadm.conf` when initializing your cluster.

You may use any CNI plugin to complete this task, but if you don't have your favourite CNI plugin's manifest URL at hand, Calico is one popular option: <https://docs.projectcalico.org/v3.14/manifests/calico.yaml>

Docker is already installed on both nodes and `apt` has been configured so that you can install the required tools.

**NEW QUESTION 24**

List all the pods sorted by created timestamp

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

```
kubect1 get pods--sort-by=.metadata.creationTimestamp
```

**NEW QUESTION 28**

Create a pod that having 3 containers in it? (Multi-Container)

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

`image=nginx`, `image=redis`, `image=consul` Name `nginx` container as `nginx-container` Name `redis` container as `redis-container` Name `consul` container as `consul-container`

Create a pod manifest file for a container and append container section for rest of the images

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

# then

```
vim multi-container.yaml apiVersion: v1
```

```
kind: Pod metadata: labels:
```

```
run: multi-container name: multi-container spec:
```

```
containers:
```

```
- image: nginx
```

name: nginx-container  
- image: redis  
name: redis-container  
- image: consul  
name: consul-container  
restartPolicy: Always

### NEW QUESTION 31

Schedule a pod as follows:

- > Name: nginx-kusc00101
- > Image: nginx
- > Node selector: disk=ssd

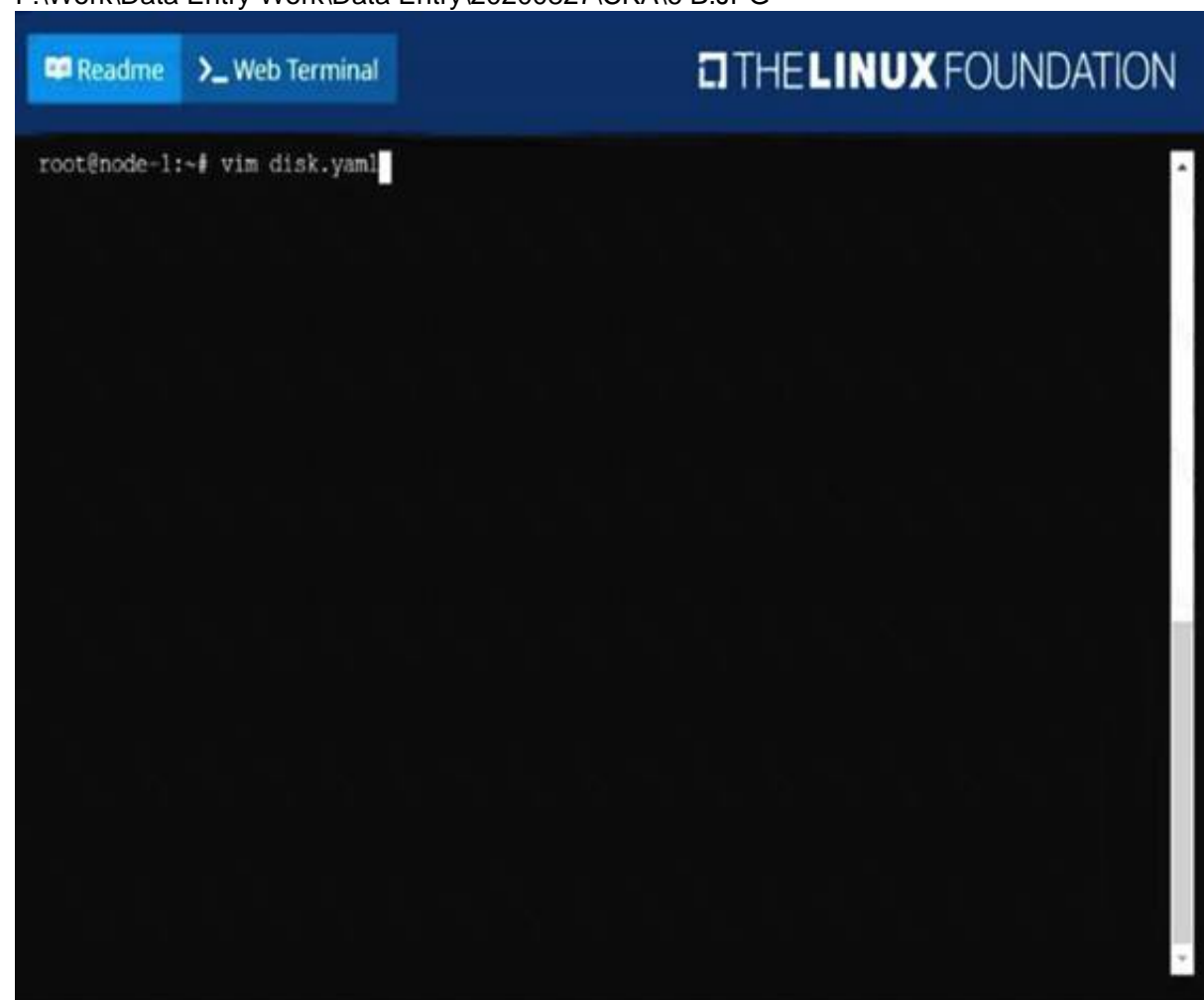
- A. Mastered  
B. Not Mastered

**Answer:** A

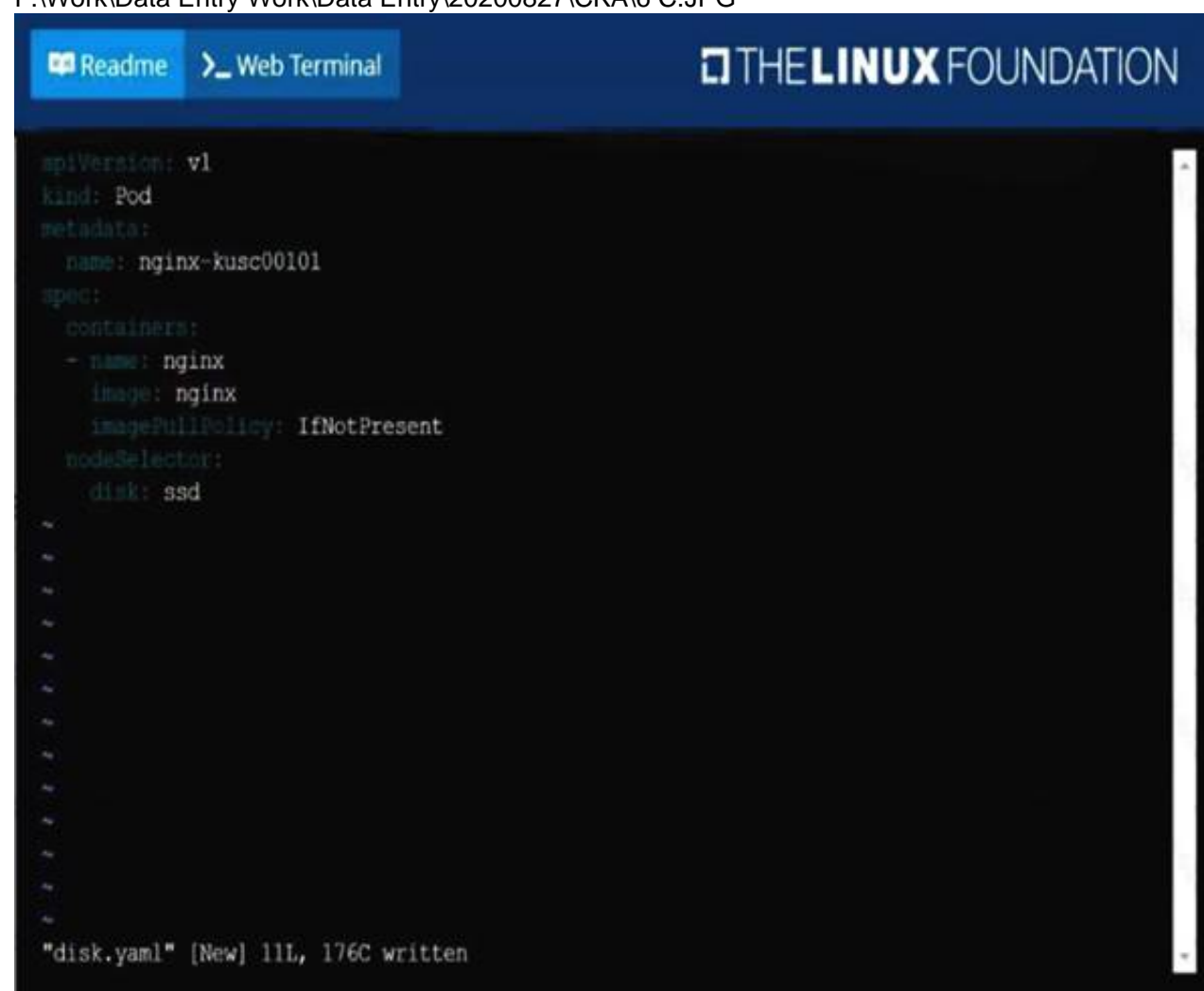
### Explanation:

solution

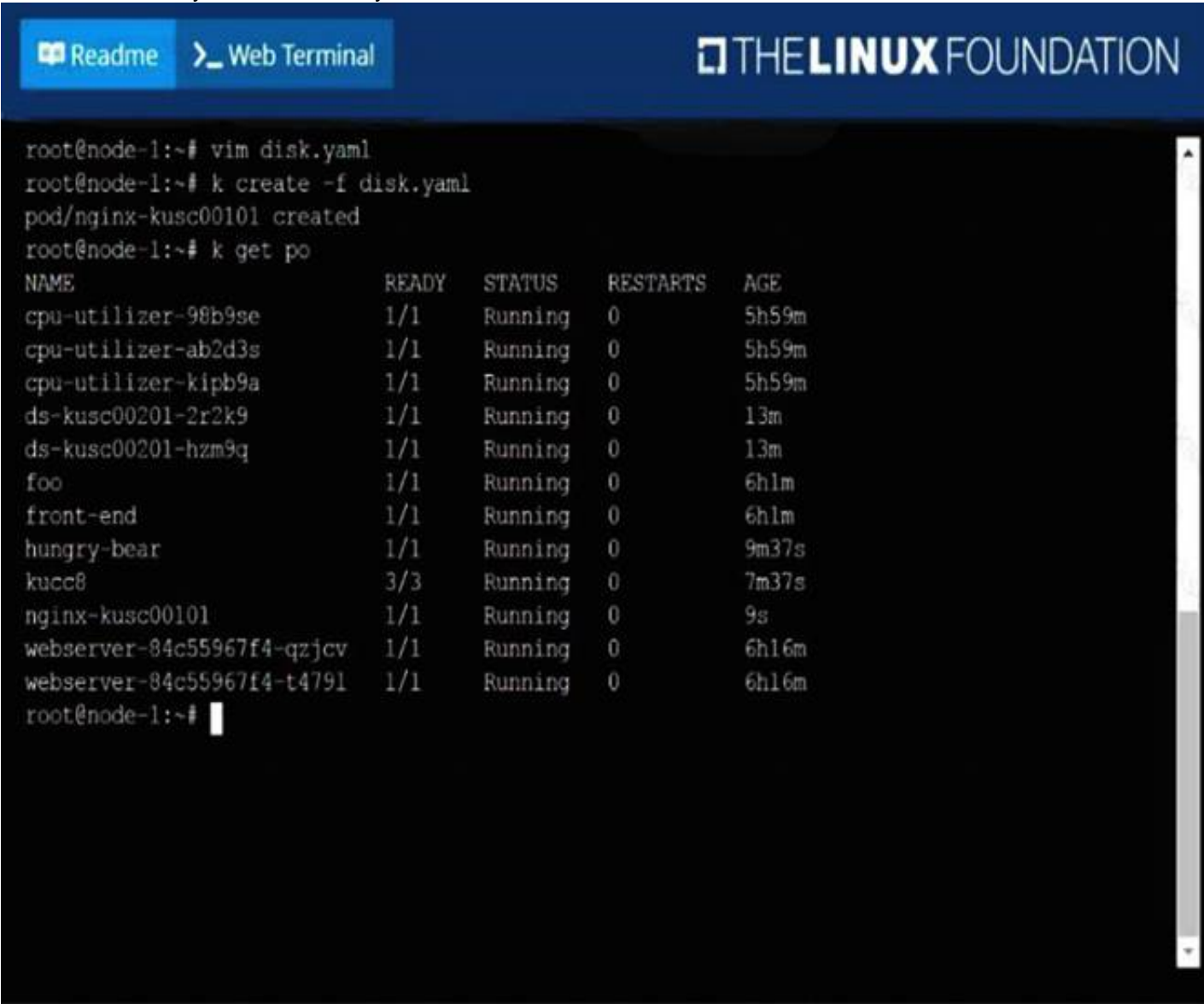
F:\Work\Data Entry Work\Data Entry\20200827\CKA\6 B.JPG



F:\Work\Data Entry Work\Data Entry\20200827\CKA\6 C.JPG



F:\Work\Data Entry Work\Data Entry\20200827\CKA\6 D.JPG



**NEW QUESTION 32**

Print pod name and start time to ??/opt/pod-status?? file

- A. Mastered
- B. Not Mastered

Answer: A

**Explanation:**

kubect1 get pods -o=jsonpath='{range items[\*]}.{metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

**NEW QUESTION 36**

Check to see how many worker nodes are ready (not including nodes taintedNoSchedule) and write the number to/opt/KUCC00104/kucc00104.txt.

- A. Mastered
- B. Not Mastered

Answer: A

**Explanation:**

solution

F:\Work\Data Entry Work\Data Entry\20200827\CKA\15 B.JPG





## Thank You for Trying Our Product

### We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

### CKA Practice Exam Features:

- \* CKA Questions and Answers Updated Frequently
- \* CKA Practice Questions Verified by Expert Senior Certified Staff
- \* CKA Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* CKA Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

**100% Actual & Verified — Instant Download, Please Click**  
**[Order The CKA Practice Test Here](#)**