

**All types of  
Certification dumps  
contact**

**WhatsApp direct link:**

<https://wa.link/e3hmdc>

**Question #:**1

**Monitor the logs of pod foo and:**

- **Extract log lines corresponding to error**  
**unable-to-access-website**
- **Write them to /opt/KULM00201/foo**

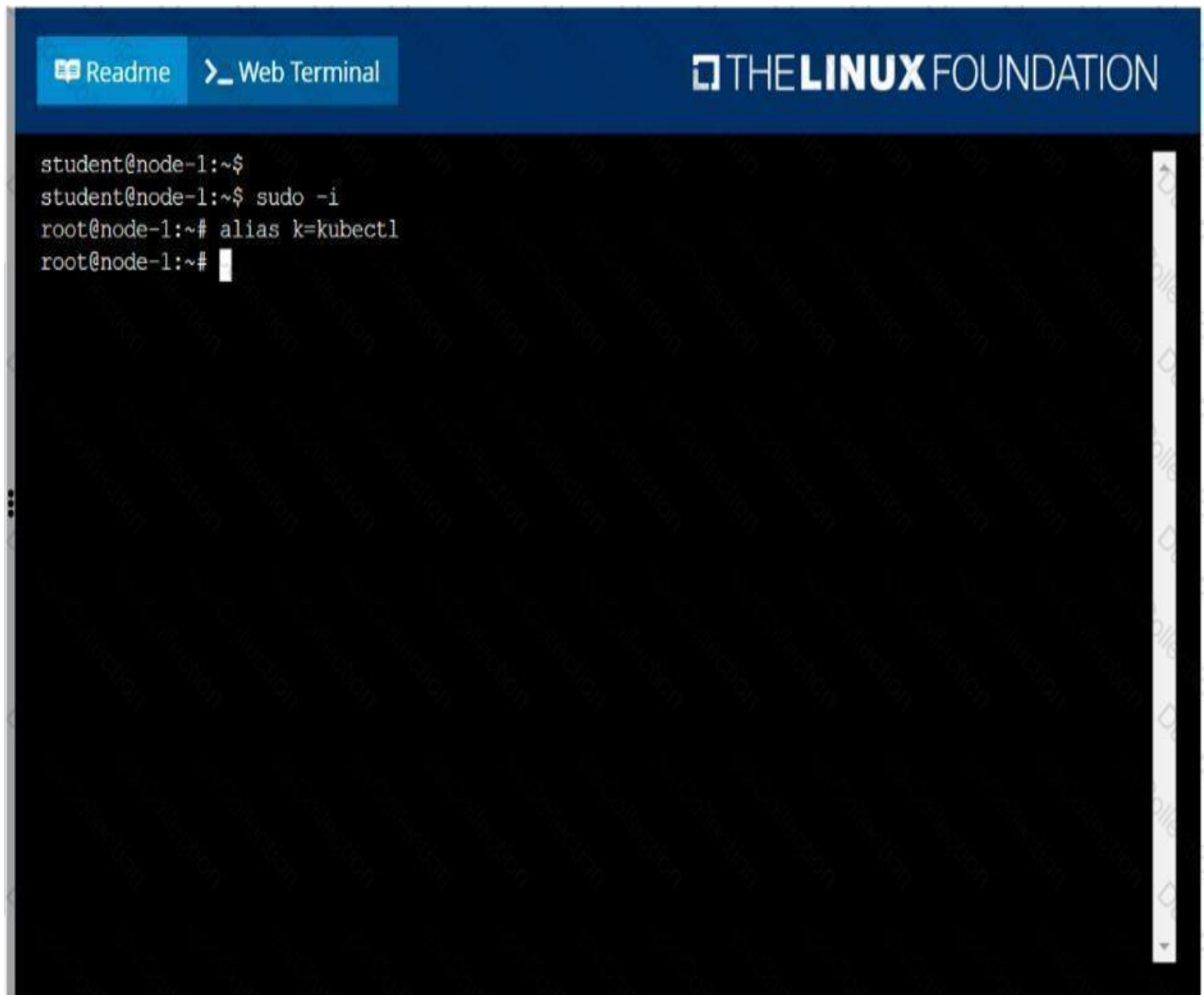


See the solution below.

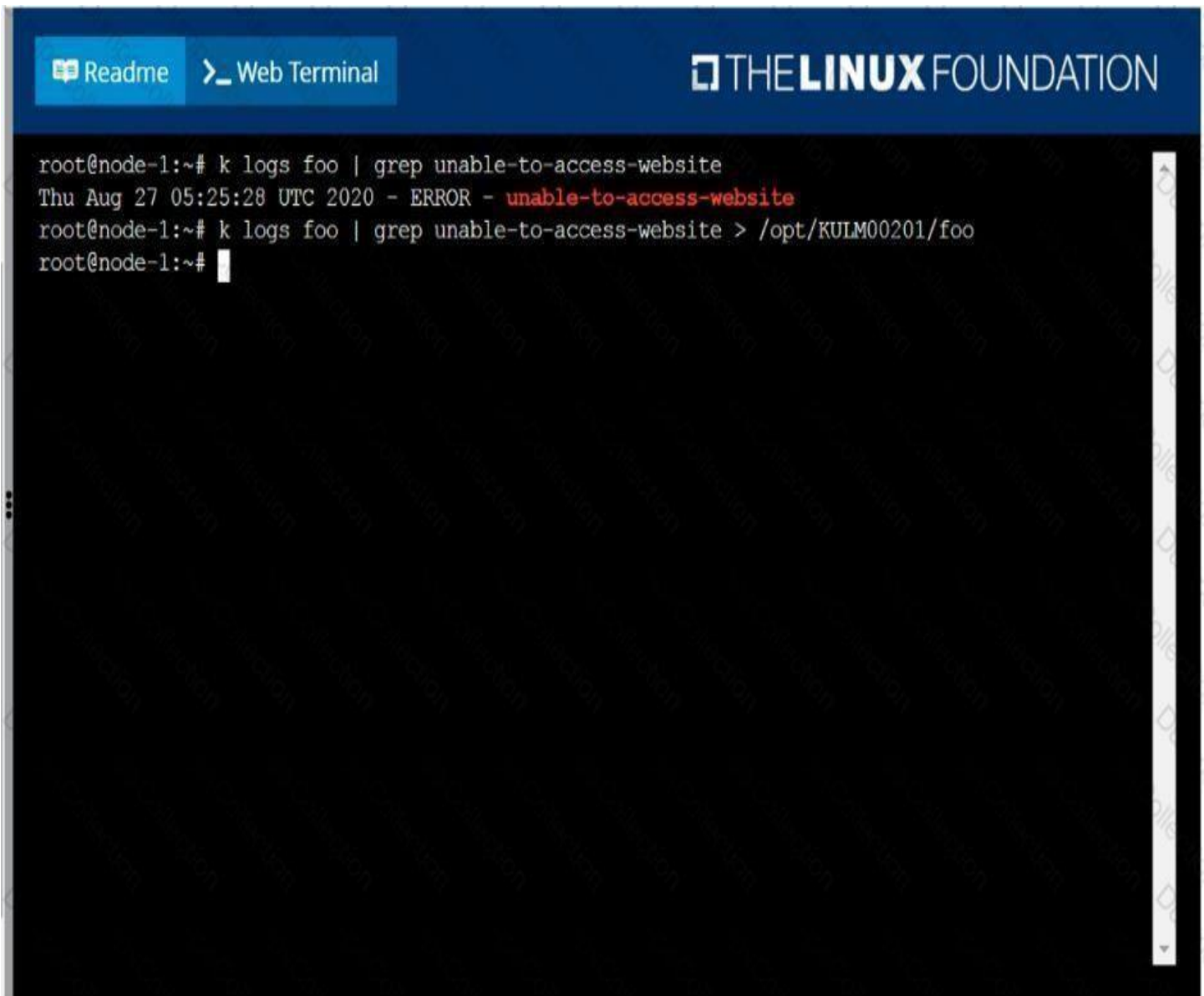
**Explanation**

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\1 B.JPG



```
student@node-1:~$  
student@node-1:~$ sudo -i  
root@node-1:~# alias k=kubectl  
root@node-1:~#
```



```
root@node-1:~# k logs foo | grep unable-to-access-website
Thu Aug 27 05:25:28 UTC 2020 - ERROR - unable-to-access-website
root@node-1:~# k logs foo | grep unable-to-access-website > /opt/KULM00201/foo
root@node-1:~#
```

### Question #2

Get IP address of the pod – “nginx-dev”

See the solution below.

### Explanation

Kubect1 get po -o wide

Using JsonPath

kubect1 get pods -o=jsonpath='{range

items[\*]}{.metadata.name}{ "\t" }{.status.podIP}{ "\n" }{end}'

**Question #3**

Create a deployment spec file that will:

- ▶ Launch 7 replicas of the `nginx` image with the label `app_runtime_stage=dev`
- ▶ deployment name: `kual00201`

Save a copy of this spec file to `/opt/KUAL00201/spec_deployment.yaml`

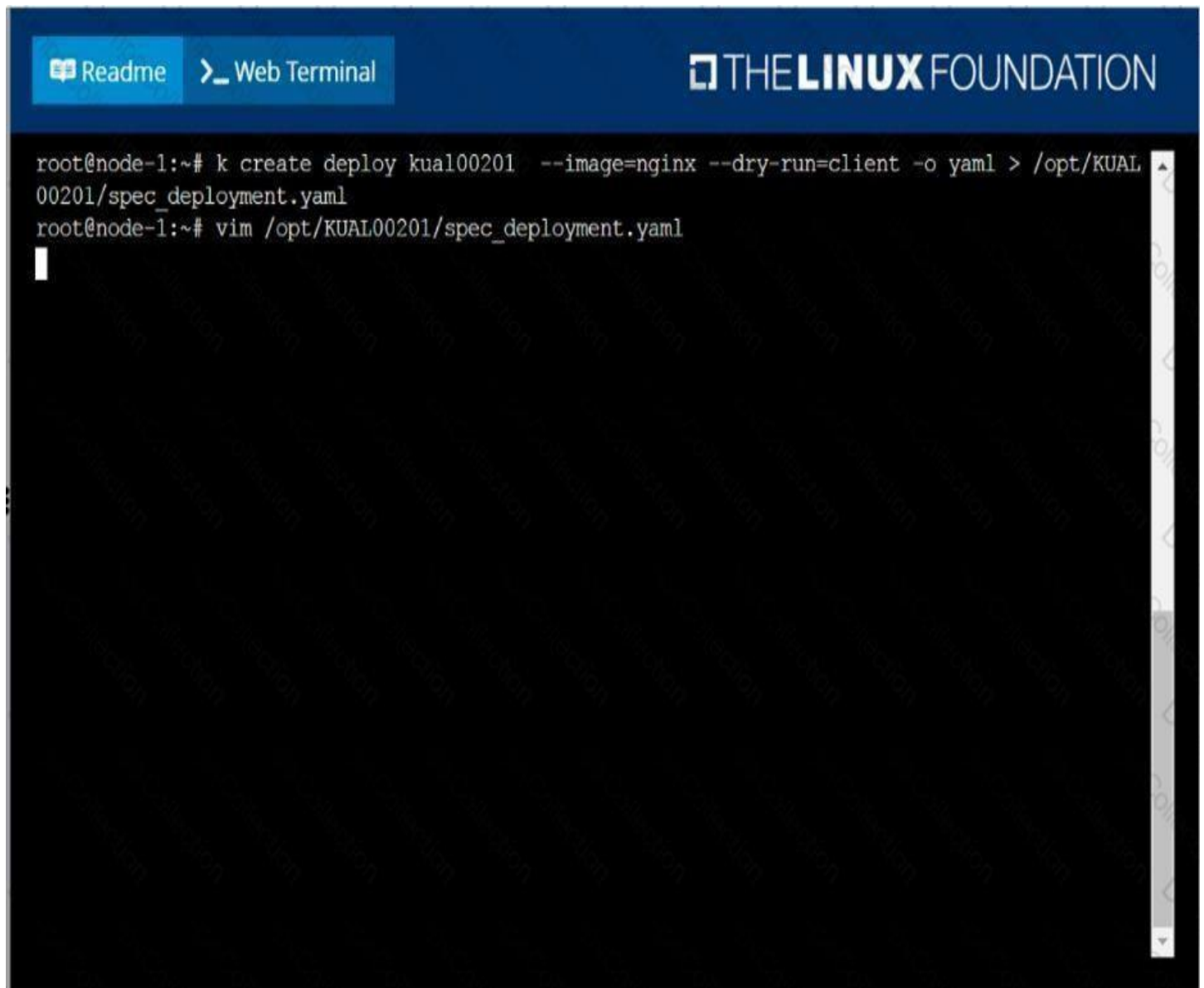
(or `/opt/KUAL00201/spec_deployment.json`).

When you are done, clean up (delete) any new Kubernetes API object that you produced during this task.

See the solution below.

**Explanation**

solution



The screenshot shows a web terminal window with a dark blue header. On the left, there are two tabs: 'Readme' and 'Web Terminal'. On the right, the 'THE LINUX FOUNDATION' logo is displayed. The terminal content shows a user at the 'root@node-1' prompt running a Kubernetes command to create a deployment named 'kual00201' with the 'nginx' image, using a dry-run client to generate a YAML file at '/opt/KUAL00201/spec\_deployment.yaml'. Following this, the user opens the file in the 'vim' editor. The editor interface is dark, with a light-colored vertical scrollbar on the right side.

```
root@node-1:~# k create deploy kual00201 --image=nginx --dry-run=client -o yaml > /opt/KUAL00201/spec_deployment.yaml
root@node-1:~# vim /opt/KUAL00201/spec_deployment.yaml
```



The screenshot shows a web terminal interface with a dark background. At the top, there are two buttons: "Readme" and "Web Terminal". The "THE LINUX FOUNDATION" logo is in the top right corner. The terminal displays a Kubernetes deployment YAML file. The file content is as follows:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app_runtime_stage: dev
  name: kual00201
spec:
  replicas: 7
  selector:
    matchLabels:
      app_runtime_stage: dev
  template:
    metadata:
      labels:
        app_runtime_stage: dev
    spec:
      containers:
      - image: nginx
        name: nginx
```

At the bottom of the terminal, a message indicates that the file has been written: `"/opt/KUAL00201/spec_deployment.yaml" 19L, 320C written`.

#### Question #4

List pod logs named “frontend” and search for the pattern “started” and write it to a file “/opt/error-logs”

See the solution below.

#### Explanation

Kubectll logs frontend | grep -i “started” > /opt/error-logs

#### Question #5

Create a Kubernetes secret asfollows:

🔍 Name: super-secret

🔍 password: bob

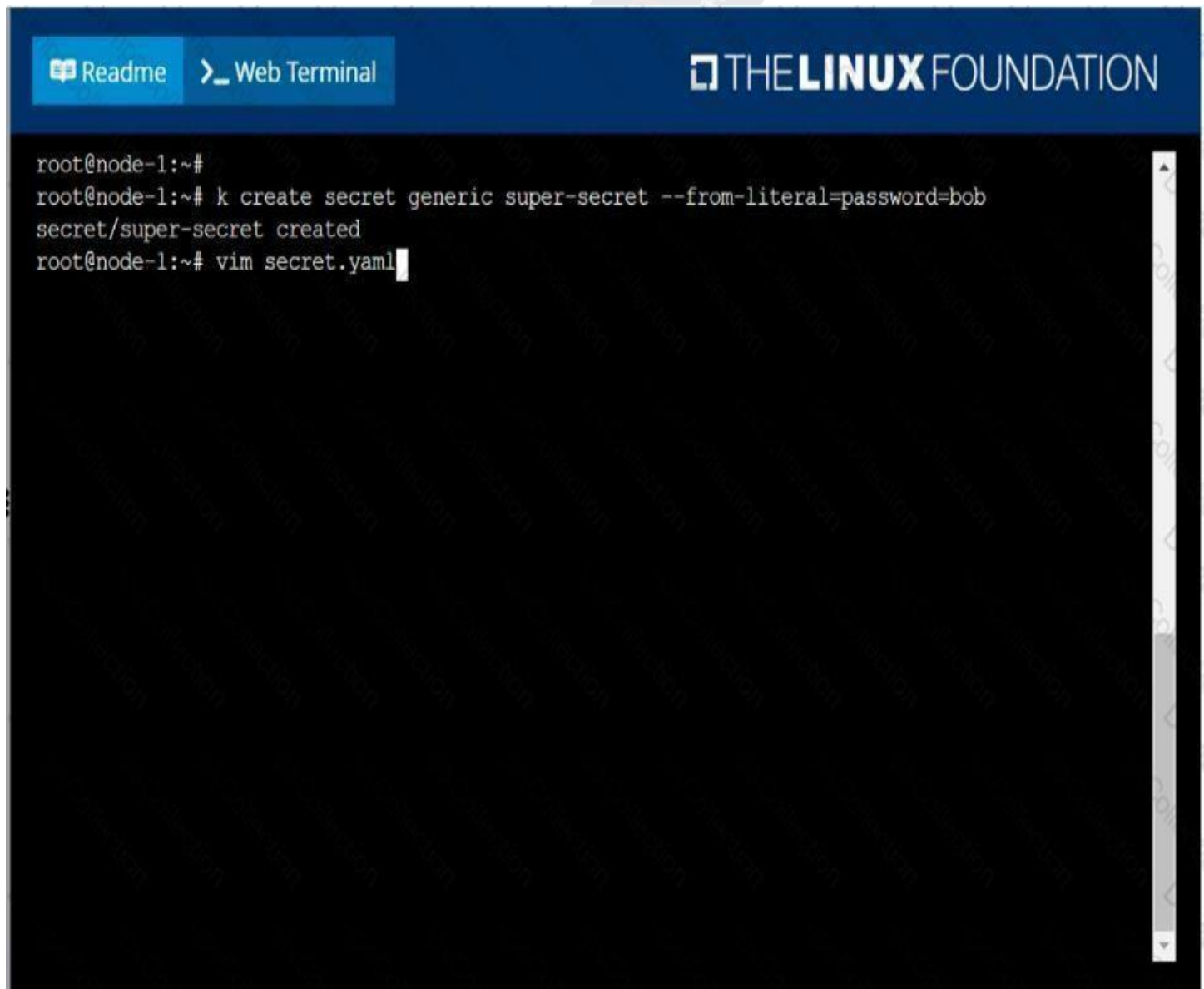
Create a pod named `pod-secrets-via-file`, using the `redisImage`, which mounts a secret named `super-secret` at `/secrets`.

Create a second pod named `pod-secrets-via-env`, using the `redisImage`, which exports `password` as `CONFIDENTIAL`

See the solution below.

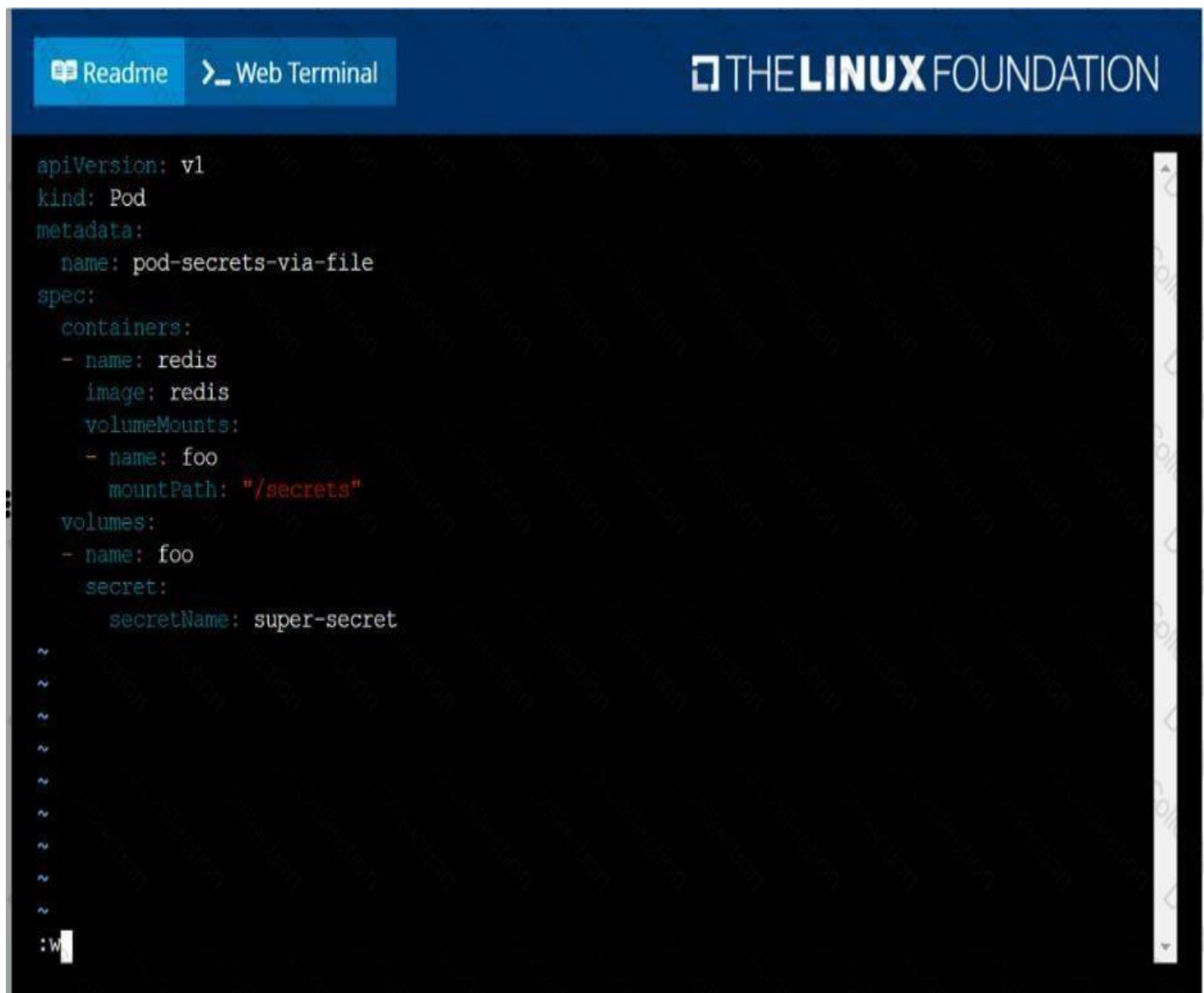
## Explanation

solution



```
root@node-1:~#  
root@node-1:~# k create secret generic super-secret --from-literal=password=bob  
secret/super-secret created  
root@node-1:~# vim secret.yaml
```






The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with two buttons: 'Readme' and 'Web Terminal'. To the right of the buttons is the 'THE LINUX FOUNDATION' logo. The main area of the terminal displays a YAML manifest for a Pod. The manifest defines a Pod named 'pod-secrets-via-file' with a single container named 'redis' using the 'redis' image. A volume named 'foo' is mounted to the container at the path '/secrets', and this volume is backed by a secret named 'super-secret'. The terminal also shows a series of tilde characters '~' and a prompt ':w' at the bottom left.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-secrets-via-file
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/secrets"
  volumes:
  - name: foo
    secret:
      secretName: super-secret

~
~
~
~
~
~
~
~
~
:w
```



The screenshot shows a terminal window with a dark background. At the top, there are two tabs: 'Readme' and 'Web Terminal'. The 'Web Terminal' tab is active. The terminal output shows the following commands and their results:

```

root@node-1:~# k create -f secret.yaml
pod/pod-secrets-via-file created
root@node-1:~# vim secret1.yaml
root@node-1:~# k create -f secret1.yaml
pod/pod-secrets-via-env created
root@node-1:~# k get po

```

Below the commands, a table of pod status is displayed:

NAME	READY	STATUS	RESTARTS	AGE
cpu-utilizer-98b9se	1/1	Running	0	6h25m
cpu-utilizer-ab2d3s	1/1	Running	0	6h25m
cpu-utilizer-kipb9a	1/1	Running	0	6h25m
ds-kusc0020l-2r2k9	1/1	Running	0	40m
ds-kusc0020l-hzm9q	1/1	Running	0	40m
foo	1/1	Running	0	6h28m
front-end	1/1	Running	0	6h27m
hungry-bear	1/1	Running	0	36m
kucc8	3/3	Running	0	34m
nginx-app-848cfcf495-9prjh	1/1	Running	0	19m
nginx-app-848cfcf495-gl2kh	1/1	Running	0	19m
nginx-app-848cfcf495-pg2c8	1/1	Running	0	19m
nginx-kusc00l0l	1/1	Running	0	26m
pod-secrets-via-env	1/1	Running	0	4s
pod-secrets-via-file	1/1	Running	0	106s
webserver-84c55967f4-qzjcv	1/1	Running	0	6h43m
webserver-84c55967f4-t479l	1/1	Running	0	6h43m

The terminal ends with a prompt: `root@node-1:~#`.

**Question #6**

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

See the solution below.

**Explanation**

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

**Question #7**

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

See the solution below.

## Explanation

```
kubectl get po -o=custom-columns="POD_NAME:.metadata.name,  
POD_STATUS:.status.containerStatuses[].state"
```

### Question #:8

Create a pod as follows:

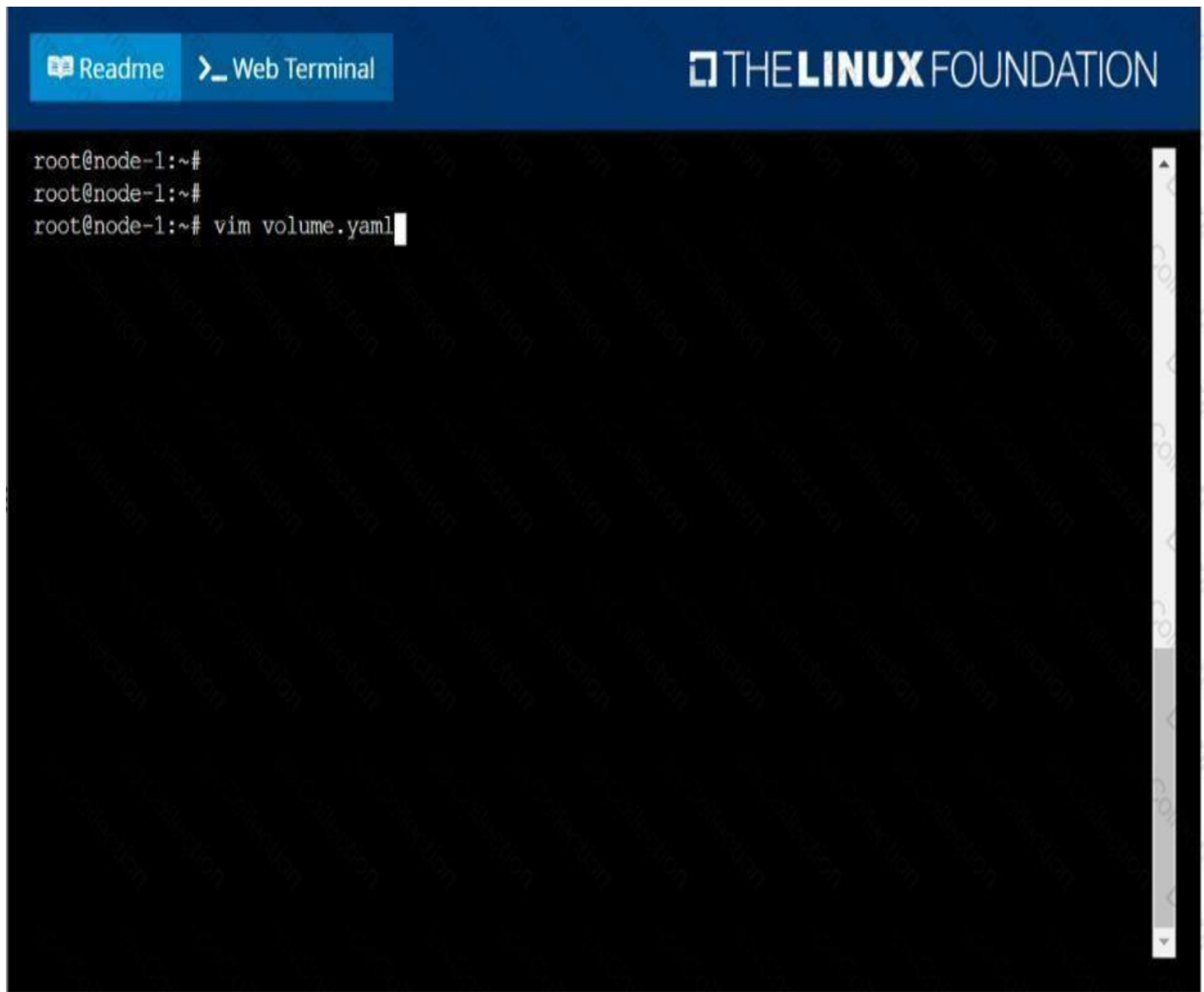
- Name:*non-persistent-redis*
- container Image:*redis*
- Volume with name:*cache-control*
- Mount path:*/data/redis*

The pod should launch in the *staging* namespace and the volume **must not** be persistent.

See the solution below.

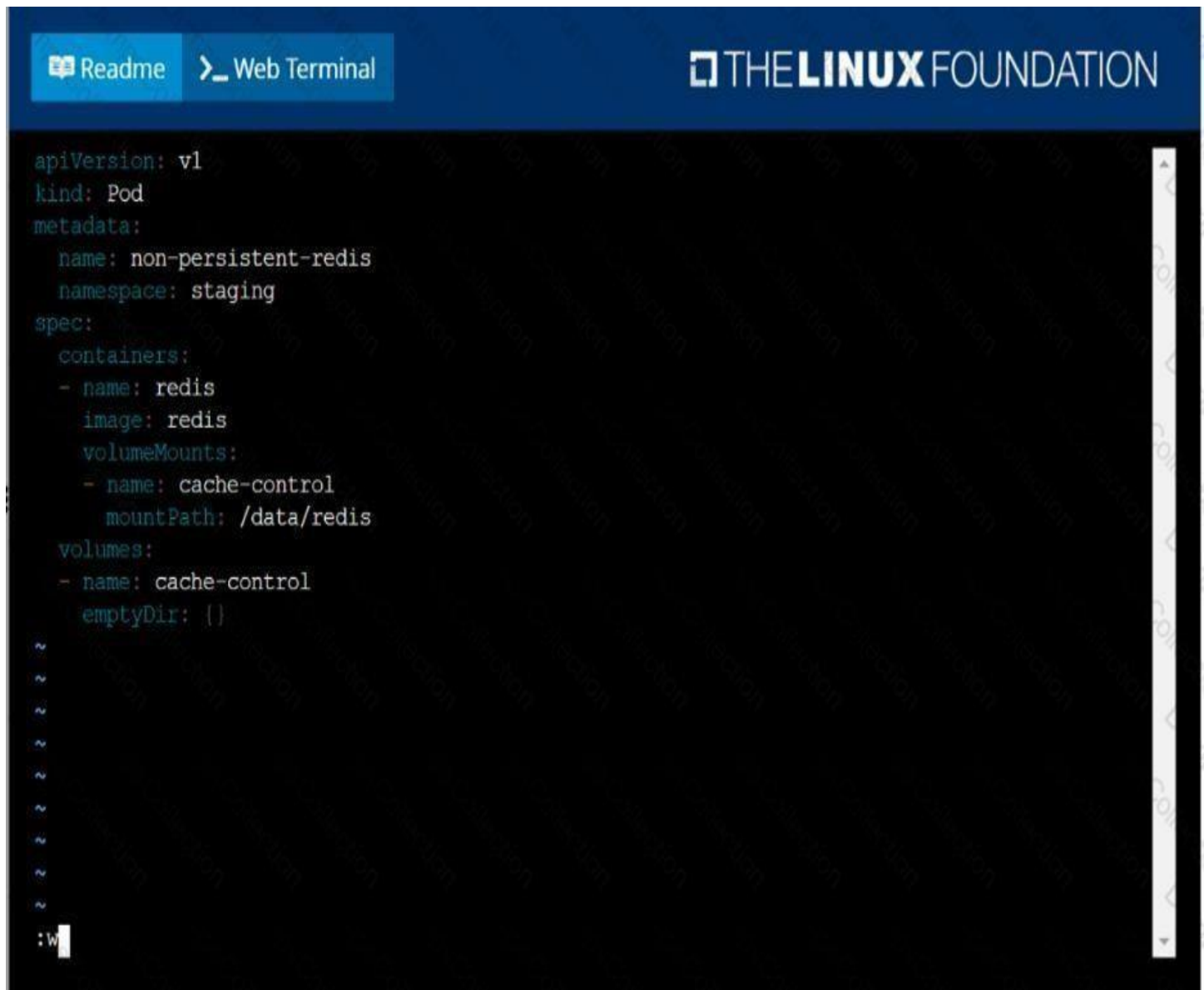
## Explanation

solution



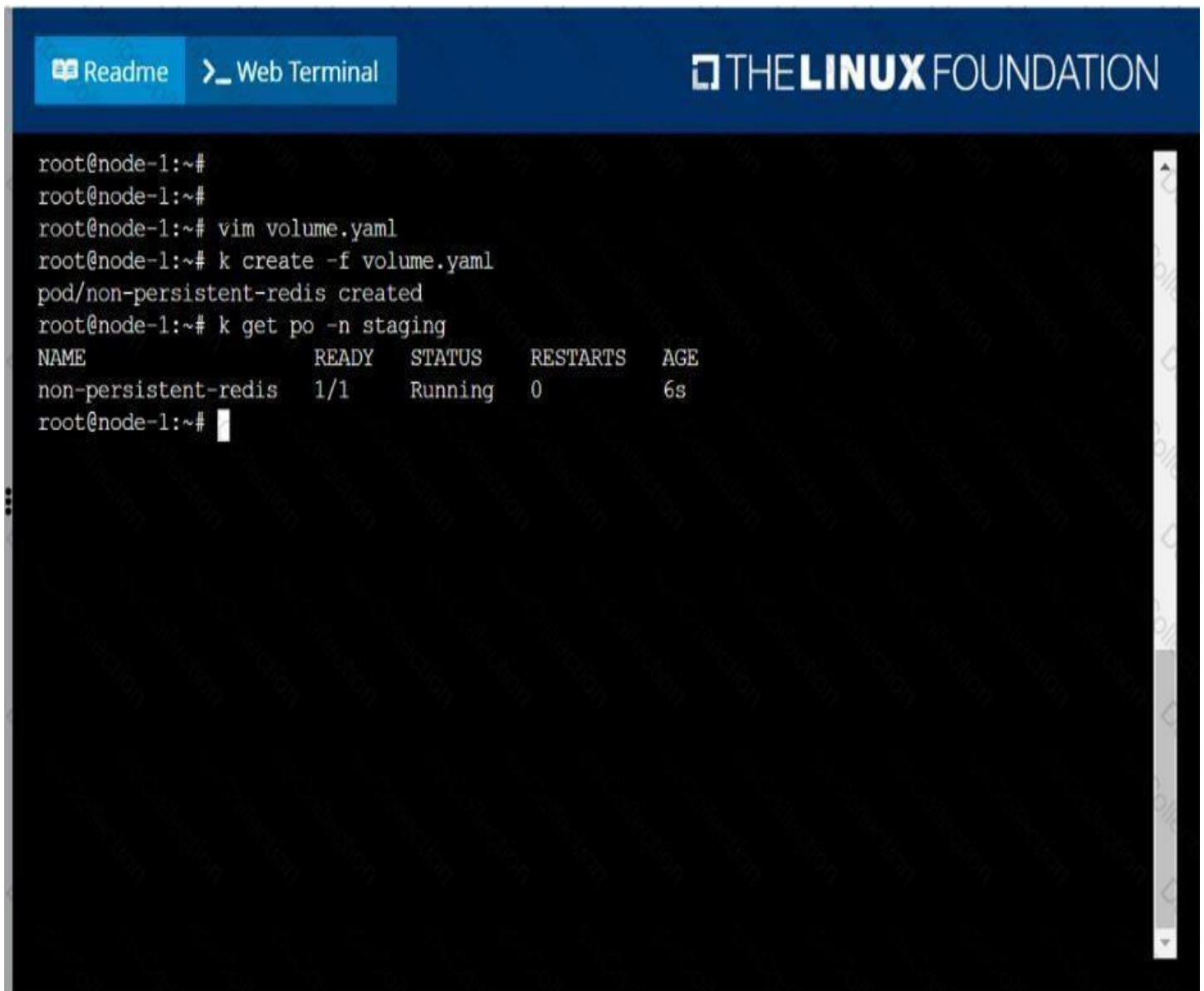
Readme Web Terminal THE **LINUX** FOUNDATION

```
root@node-1:~#  
root@node-1:~#  
root@node-1:~# vim volume.yaml
```



The screenshot shows a web terminal interface with a dark background. At the top, there are two tabs: 'Readme' and 'Web Terminal'. The 'Web Terminal' tab is active. In the top right corner, the 'THE LINUX FOUNDATION' logo is displayed. The terminal content shows a Kubernetes pod specification for a Redis pod. The specification includes the API version, kind, namespace, and container details. The container is named 'redis' and uses the 'redis' image. It has a volume mount for 'cache-control' at the path '/data/redis'. The volume is named 'cache-control' and is an empty directory. The terminal also shows a prompt ':w' at the bottom left.

```
apiVersion: v1
kind: Pod
metadata:
  name: non-persistent-redis
  namespace: staging
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: cache-control
      mountPath: /data/redis
  volumes:
  - name: cache-control
    emptyDir: {}
~
~
~
~
~
~
~
~
:w
```



The screenshot shows a web terminal window with a dark background. At the top, there is a blue header bar with a 'Readme' button and a 'Web Terminal' button. The terminal text shows a user at 'root@node-1:~#'. The user runs 'vim volume.yaml', then 'k create -f volume.yaml', which outputs 'pod/non-persistent-redis created'. Then the user runs 'k get po -n staging', which outputs a table of pod information.

```
root@node-1:~#  
root@node-1:~#  
root@node-1:~# vim volume.yaml  
root@node-1:~# k create -f volume.yaml  
pod/non-persistent-redis created  
root@node-1:~# k get po -n staging  
NAME                READY   STATUS    RESTARTS   AGE  
non-persistent-redis 1/1     Running   0           6s  
root@node-1:~#
```

### Question #:9

Create a deployment as follows:

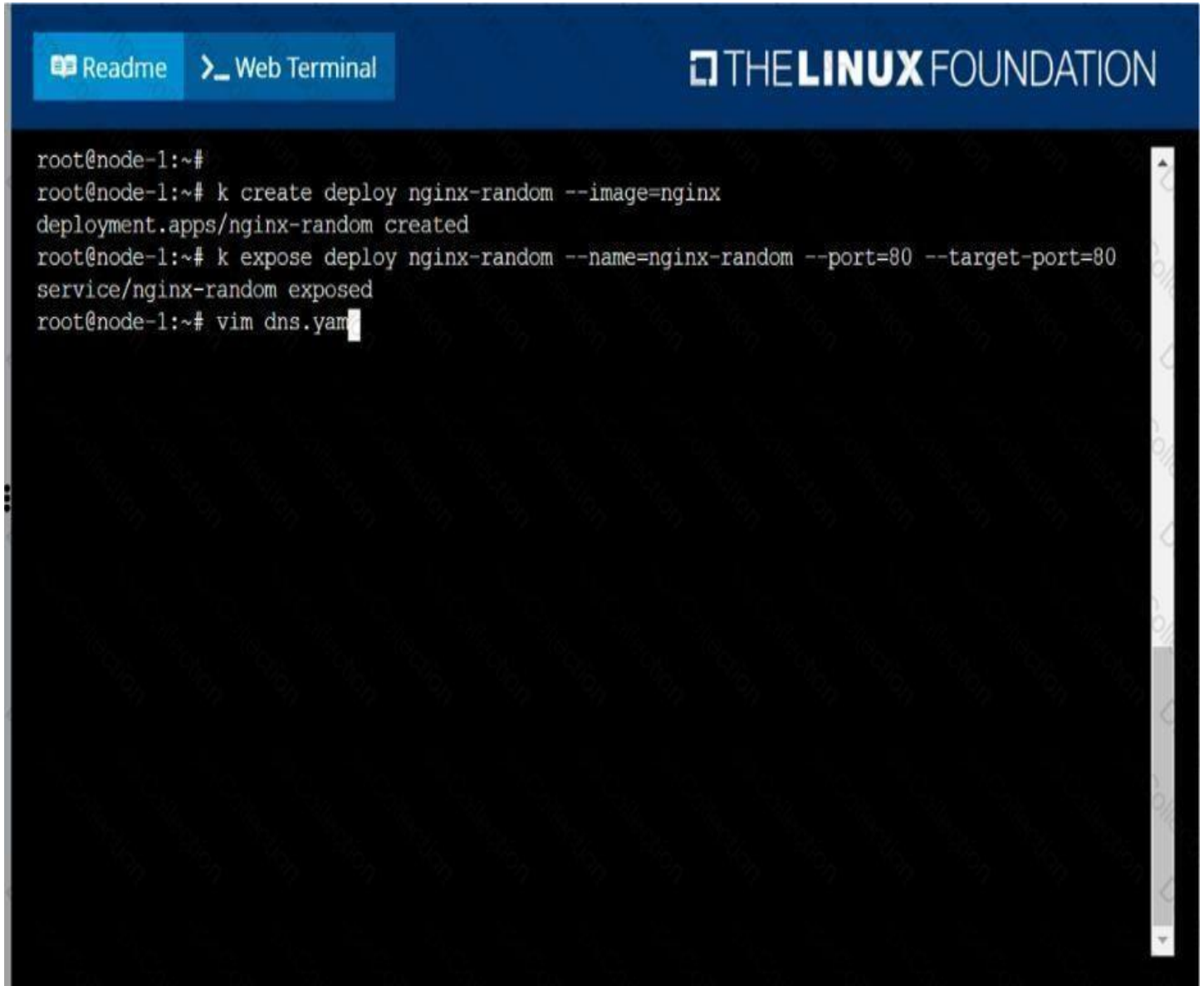
- Name: *nginx-random*
- Exposed via a service *nginx-random*
- Ensure that the service & pod are accessible via their respective DNS records
- The container(s) within any pod(s) running as a part of this deployment should use the *nginx* image

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.

See the solution below.

## Explanation

Solution:

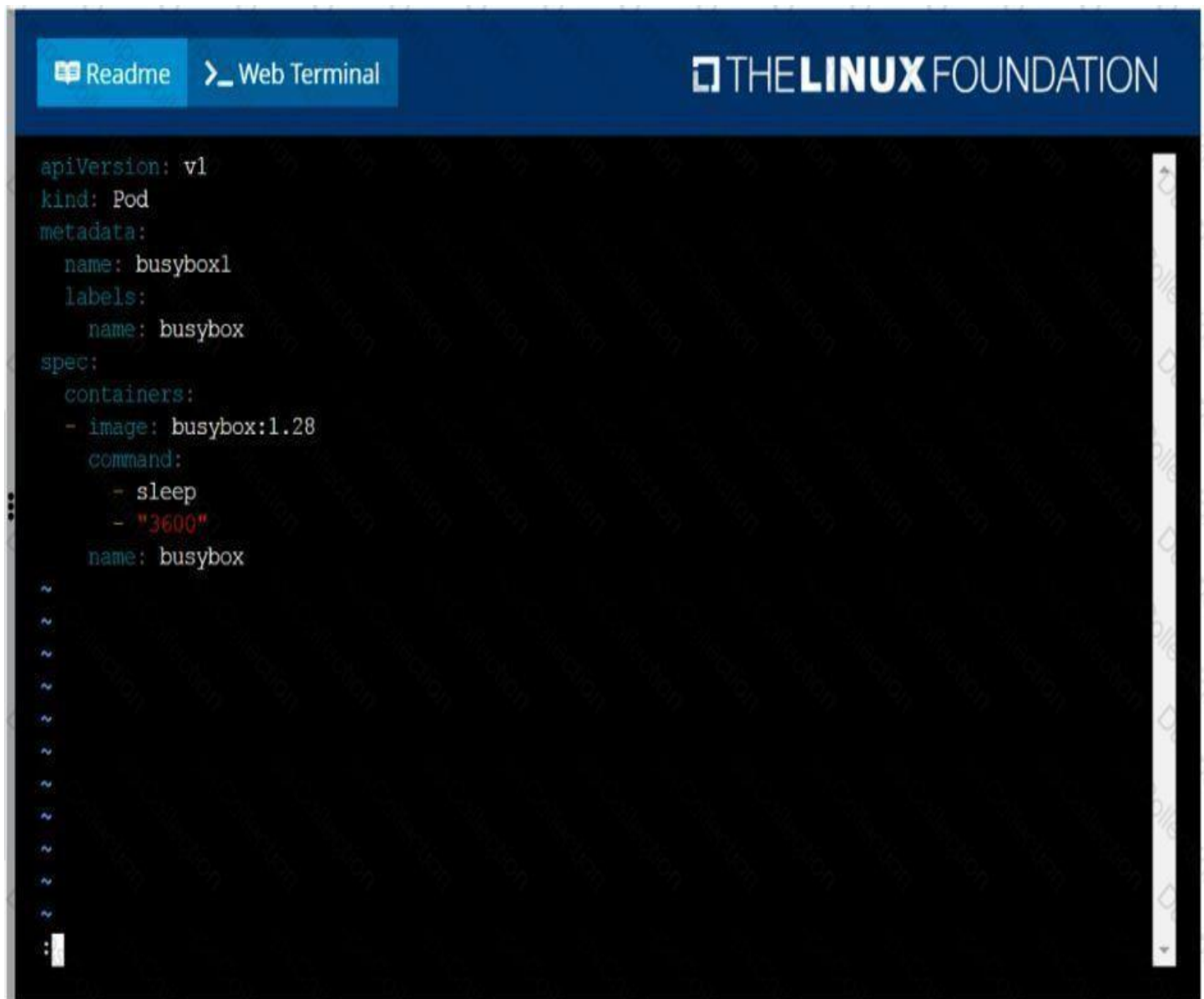


The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with two buttons on the left: 'Readme' (with a document icon) and 'Web Terminal' (with a terminal icon). On the right side of the header is the 'THE LINUX FOUNDATION' logo. The terminal window displays the following text:

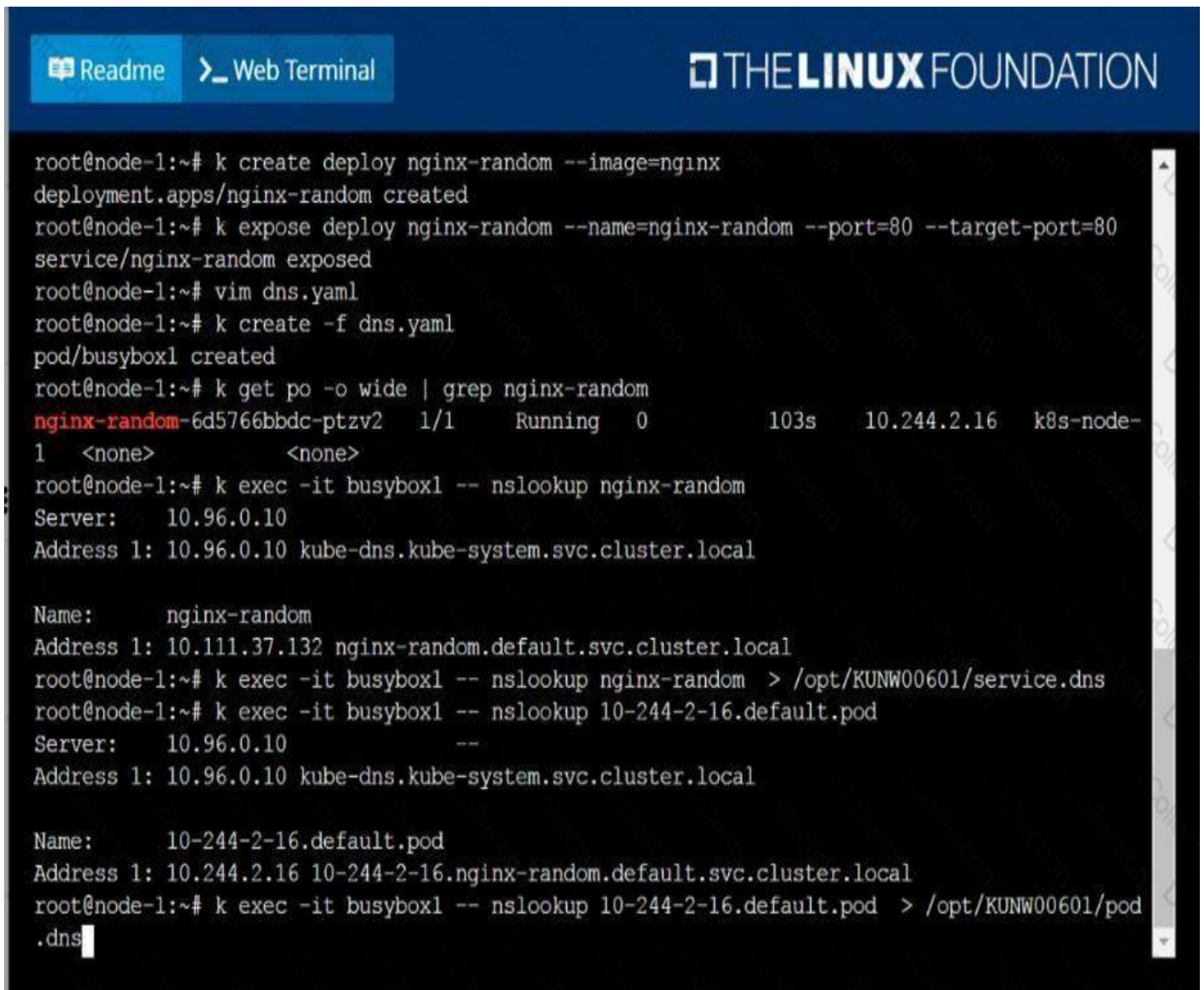
```
root@node-1:~#  
root@node-1:~# k create deploy nginx-random --image=nginx  
deployment.apps/nginx-random created  
root@node-1:~# k expose deploy nginx-random --name=nginx-random --port=80 --target-port=80  
service/nginx-random exposed  
root@node-1:~# vim dns.yaml
```

The terminal text is white on a black background. A vertical scrollbar is visible on the right side of the terminal window.









```
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
```

#### Question #:10

List all the pods sorted by name

See the solution below.

#### Explanation

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

#### Question #:11

Create a deployment as follows:

- Name:*nginx-app*
- Using container*nginx*withversion 1.11.10-alpine
- The deployment should contain3replicas

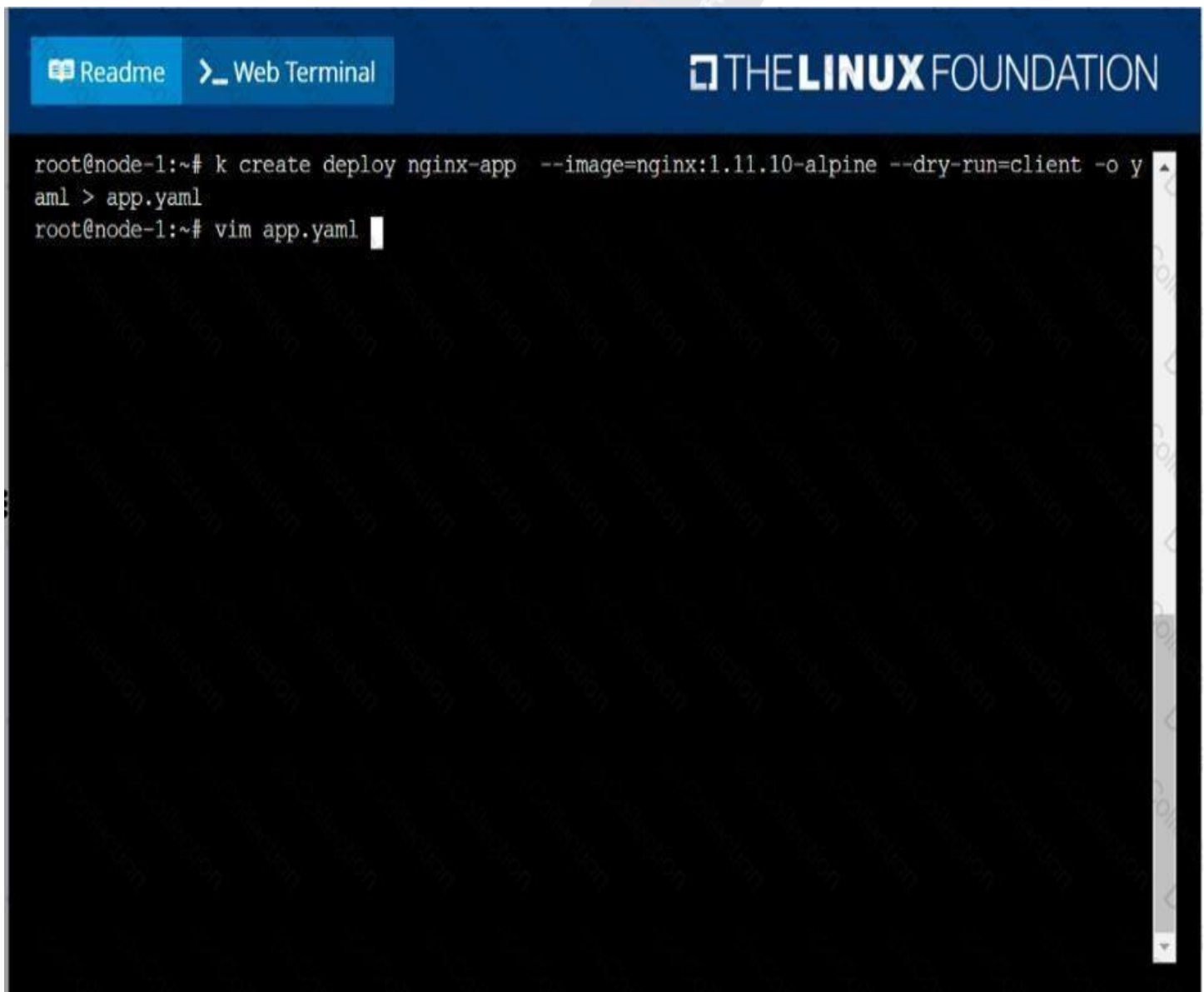
Next, deploy the application with newversion*1.11.13-alpine*, byperforming a rolling update.

Finally, rollback that update to theprevious version*1.11.10-alpine*.

See the solution below.

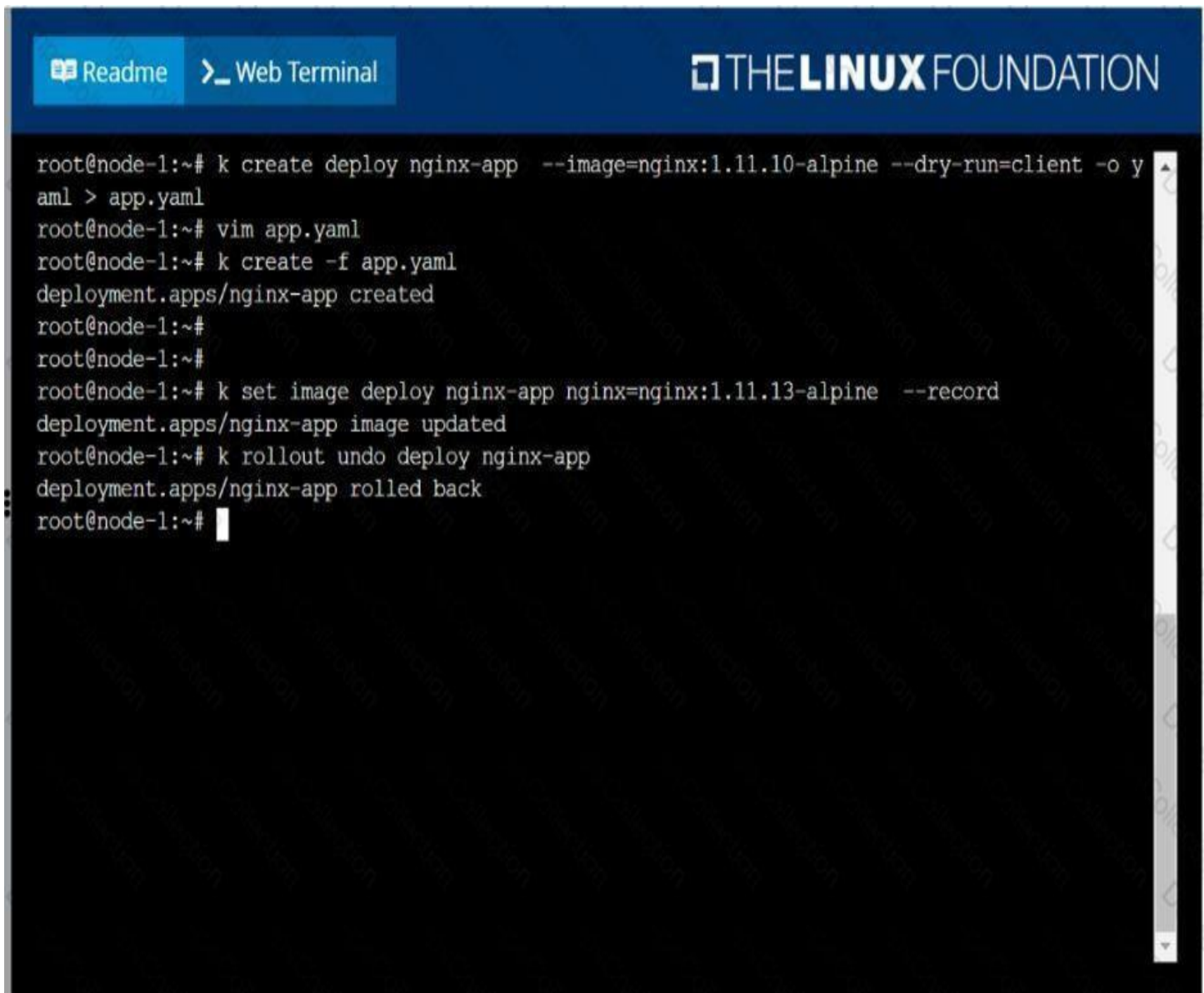
## Explanation

solution



```
root@node-1:~# k create deploy nginx-app --image=nginx:1.11.10-alpine --dry-run=client -o y
aml > app.yaml
root@node-1:~# vim app.yaml
```





```
root@node-1:~# k create deploy nginx-app --image=nginx:1.11.10-alpine --dry-run=client -o y
aml > app.yaml
root@node-1:~# vim app.yaml
root@node-1:~# k create -f app.yaml
deployment.apps/nginx-app created
root@node-1:~#
root@node-1:~#
root@node-1:~# k set image deploy nginx-app nginx=nginx:1.11.13-alpine --record
deployment.apps/nginx-app image updated
root@node-1:~# k rollout undo deploy nginx-app
deployment.apps/nginx-app rolled back
root@node-1:~#
```

### Question #:12

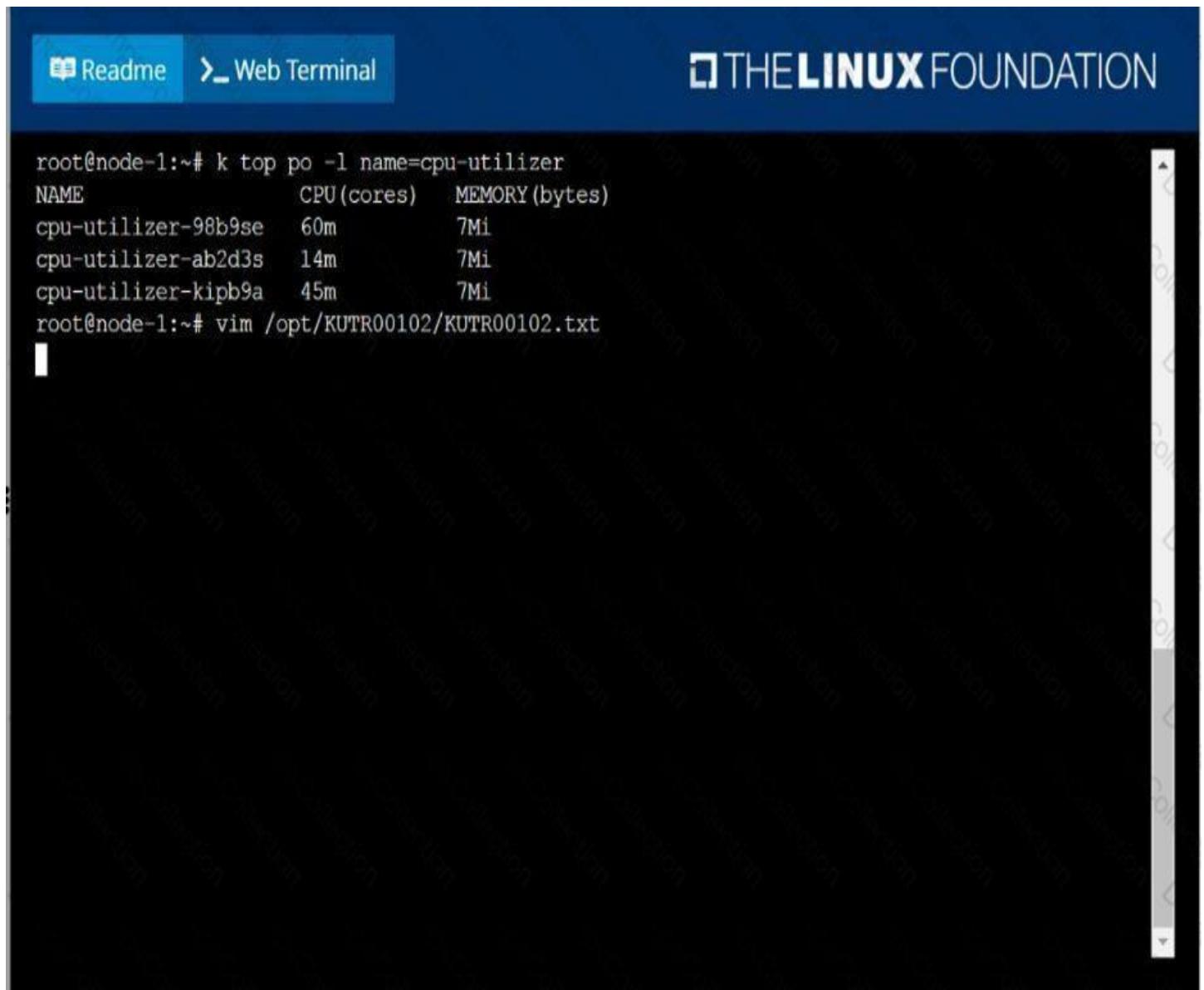
From the pod `labelname=cpu-utilizer`, find pods running high CPU workloads and

write the name of the pod consuming most CPU to the file `/opt/KUTR00102/KUTR00102.txt` (which already exists).

See the solution below.

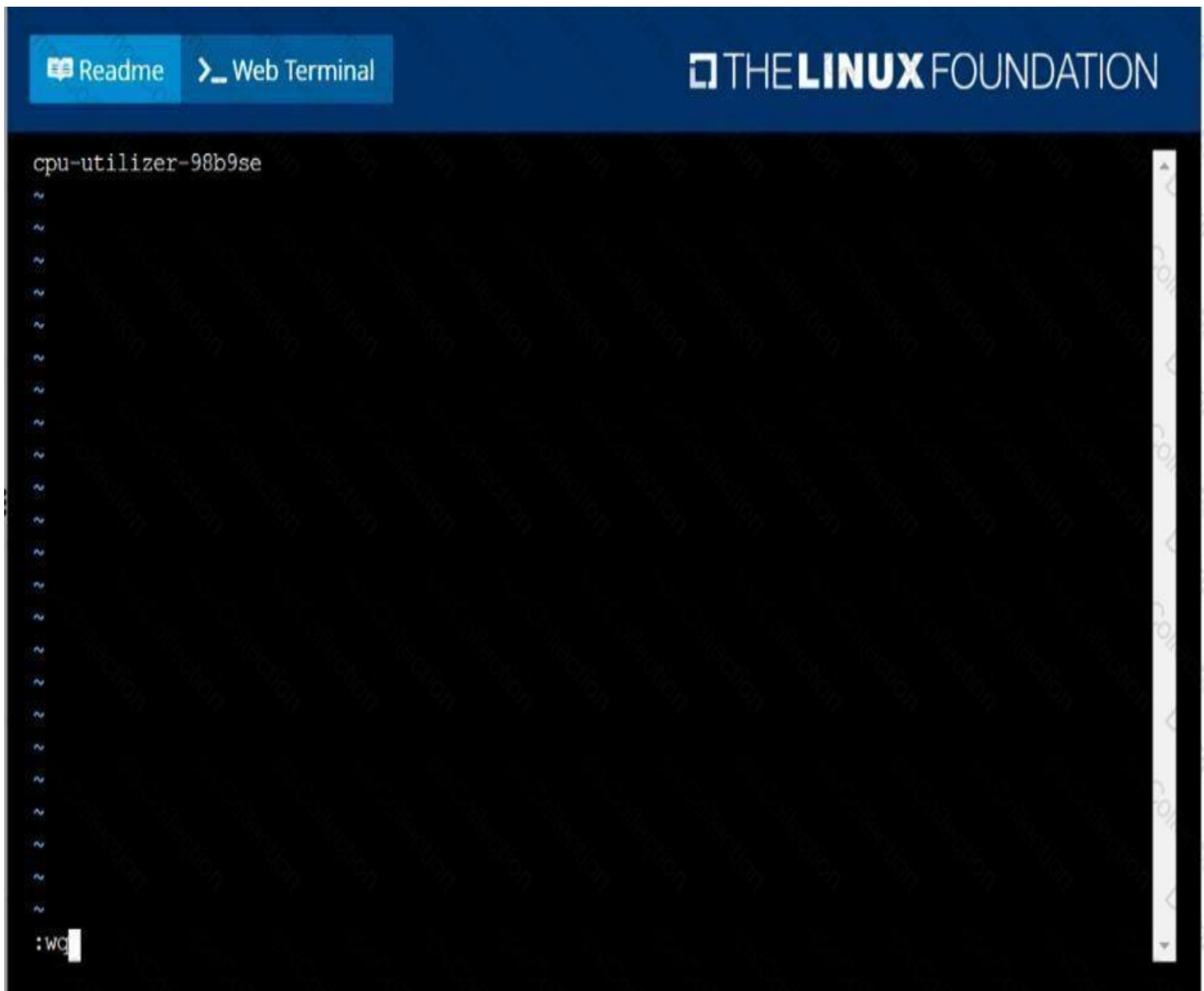
### Explanation

solution



Readme Web Terminal THE **LINUX** FOUNDATION

```
root@node-1:~# k top po -l name=cpu-utilizer
NAME                CPU(cores)  MEMORY(bytes)
cpu-utilizer-98b9se  60m         7Mi
cpu-utilizer-ab2d3s  14m         7Mi
cpu-utilizer-kipb9a  45m         7Mi
root@node-1:~# vim /opt/KUTR00102/KUTR00102.txt
█
```



### Question #:13

Create a snapshot of the `etcd` instance running at `https://127.0.0.1:2379`, saving the snapshot to the file path `/srv/data/etcd-snapshot.db`.

The following TLS certificates/key are supplied for connecting to the server with `etcdctl`:

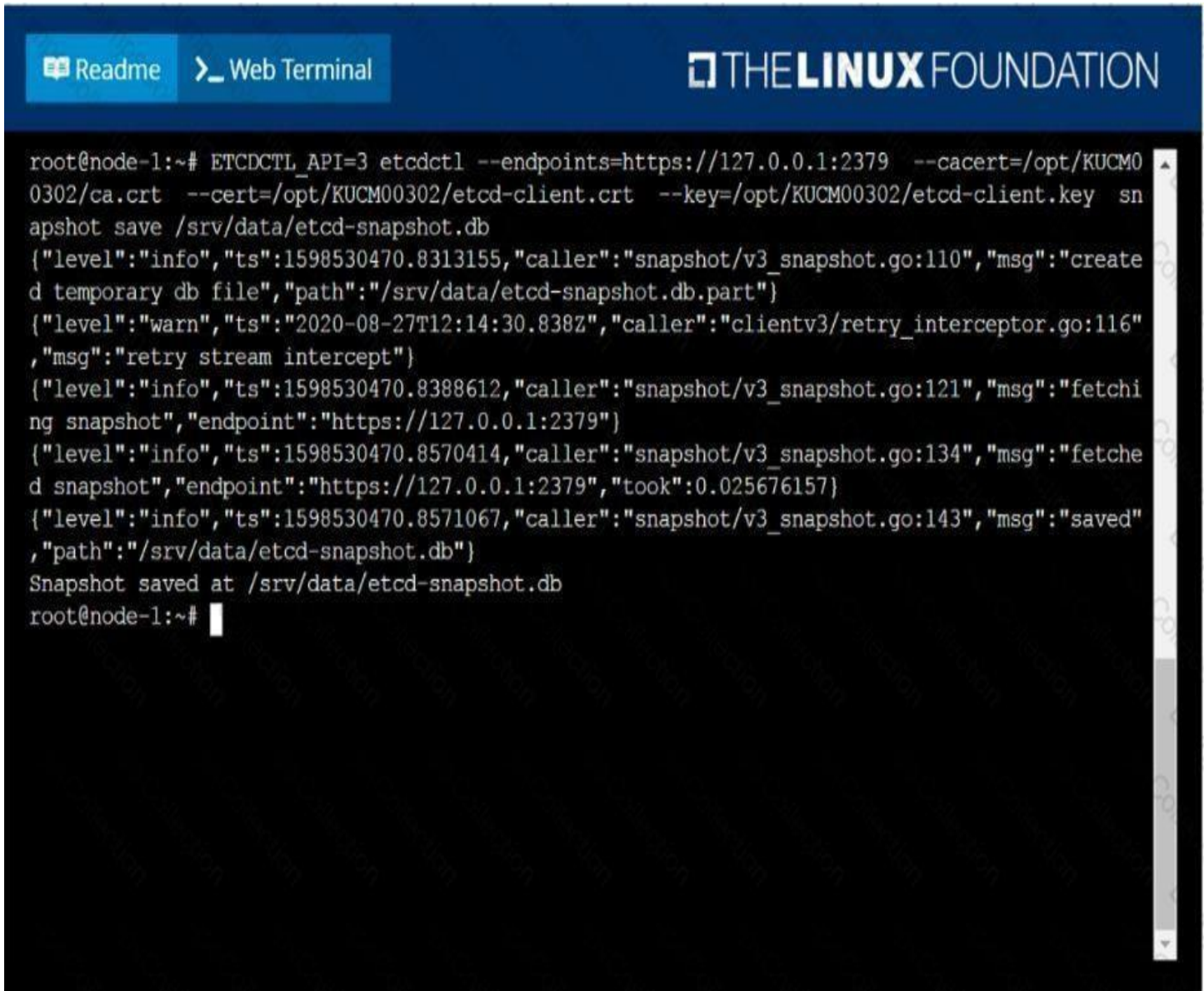
- CA certificate: `/opt/KUCM00302/ca.crt`
- Client certificate: `/opt/KUCM00302/etcd-client.crt`
- Client key: `/opt/KUCM00302/etcd-client.key`

See the solution below.

### Explanation



solution



```
root@node-1:~# ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/opt/KUCM00302/ca.crt --cert=/opt/KUCM00302/etcd-client.crt --key=/opt/KUCM00302/etcd-client.key snapshot save /srv/data/etcd-snapshot.db
{"level":"info","ts":1598530470.8313155,"caller":"snapshot/v3_snapshot.go:110","msg":"create temporary db file","path":"/srv/data/etcd-snapshot.db.part"}
{"level":"warn","ts":"2020-08-27T12:14:30.838Z","caller":"clientv3/retry_interceptor.go:116","msg":"retry stream intercept"}
{"level":"info","ts":1598530470.8388612,"caller":"snapshot/v3_snapshot.go:121","msg":"fetching snapshot","endpoint":"https://127.0.0.1:2379"}
{"level":"info","ts":1598530470.8570414,"caller":"snapshot/v3_snapshot.go:134","msg":"fetched snapshot","endpoint":"https://127.0.0.1:2379","took":0.025676157}
{"level":"info","ts":1598530470.8571067,"caller":"snapshot/v3_snapshot.go:143","msg":"saved","path":"/srv/data/etcd-snapshot.db"}
Snapshot saved at /srv/data/etcd-snapshot.db
root@node-1:~#
```

#### Question #:14

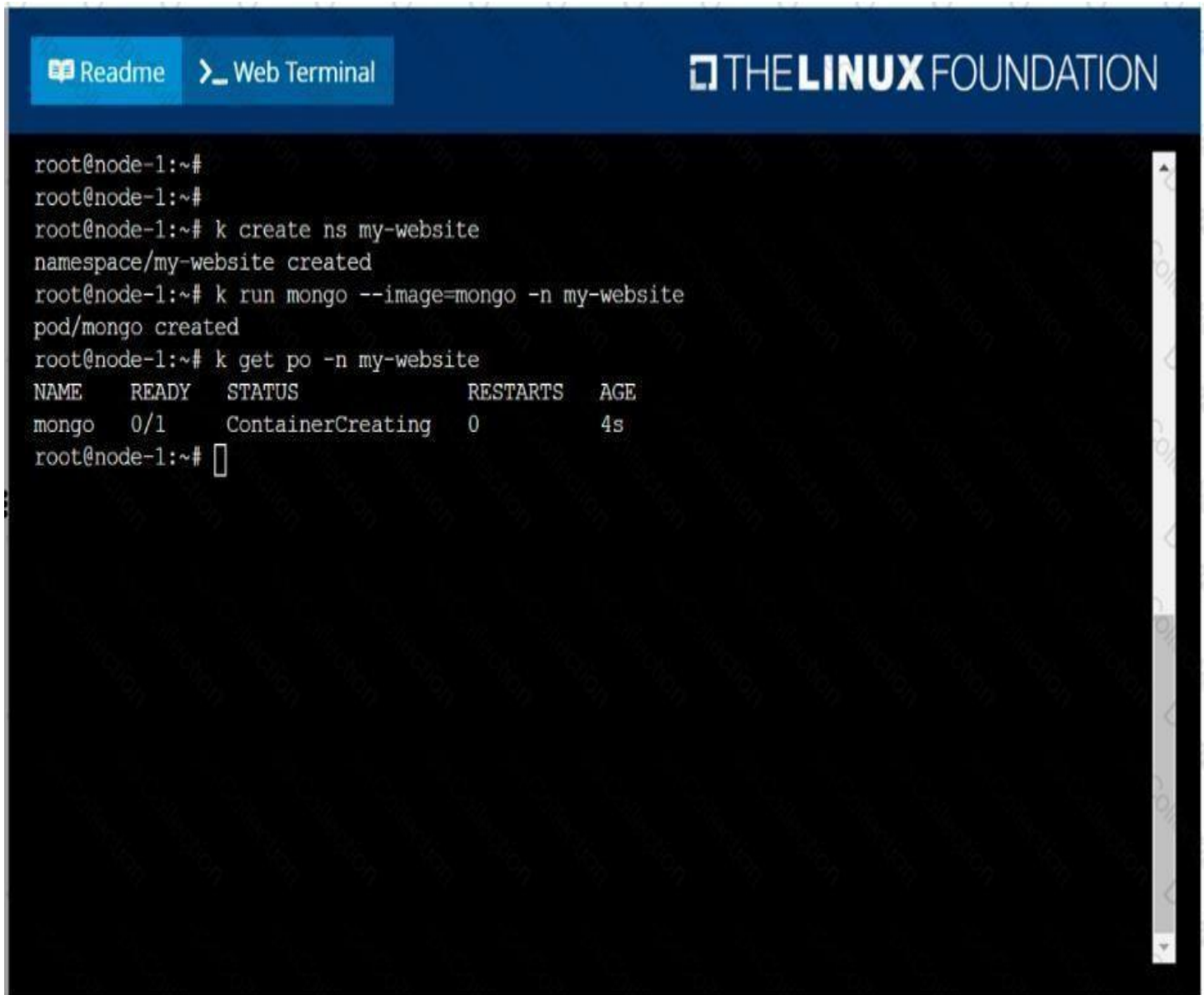
Create a pod as follows:

- Name:*mongo*
- Using Image:*mongo*
- In anew Kubernetes namespace named:*my-website*

See the solution below.

## Explanation

solution



```
root@node-1:~#  
root@node-1:~#  
root@node-1:~# k create ns my-website  
namespace/my-website created  
root@node-1:~# k run mongo --image=mongo -n my-website  
pod/mongo created  
root@node-1:~# k get po -n my-website  
NAME      READY   STATUS             RESTARTS   AGE  
mongo     0/1     ContainerCreating   0           4s  
root@node-1:~#
```

### Question #:15

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.

See the solution below.

## Explanation

```
kubectl 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
```

```
# kubectl create -f nginx-prod-pod.yaml
```

```
kubectl 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

# kubectl create -f nginx-prod-dev.yaml

Verify :

kubectl get po --show-labels

kubectl get po -l env=prod

kubectl get po -l env=dev
```

### Question #:16

**Schedule a pod as follows:**

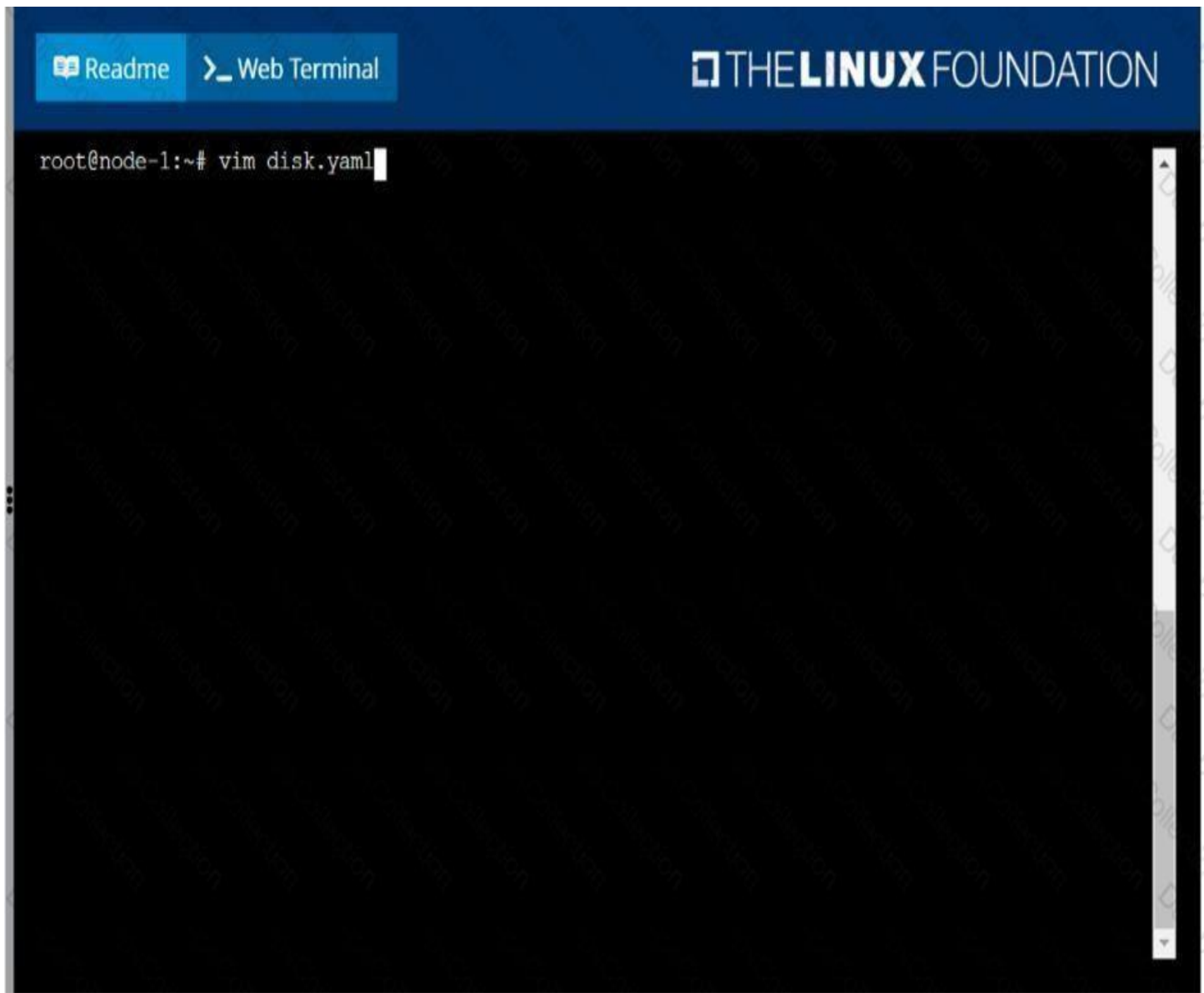
- **Name: nginx-kusc00101**
- **Image: nginx**
- **Node selector: disk=ssd**

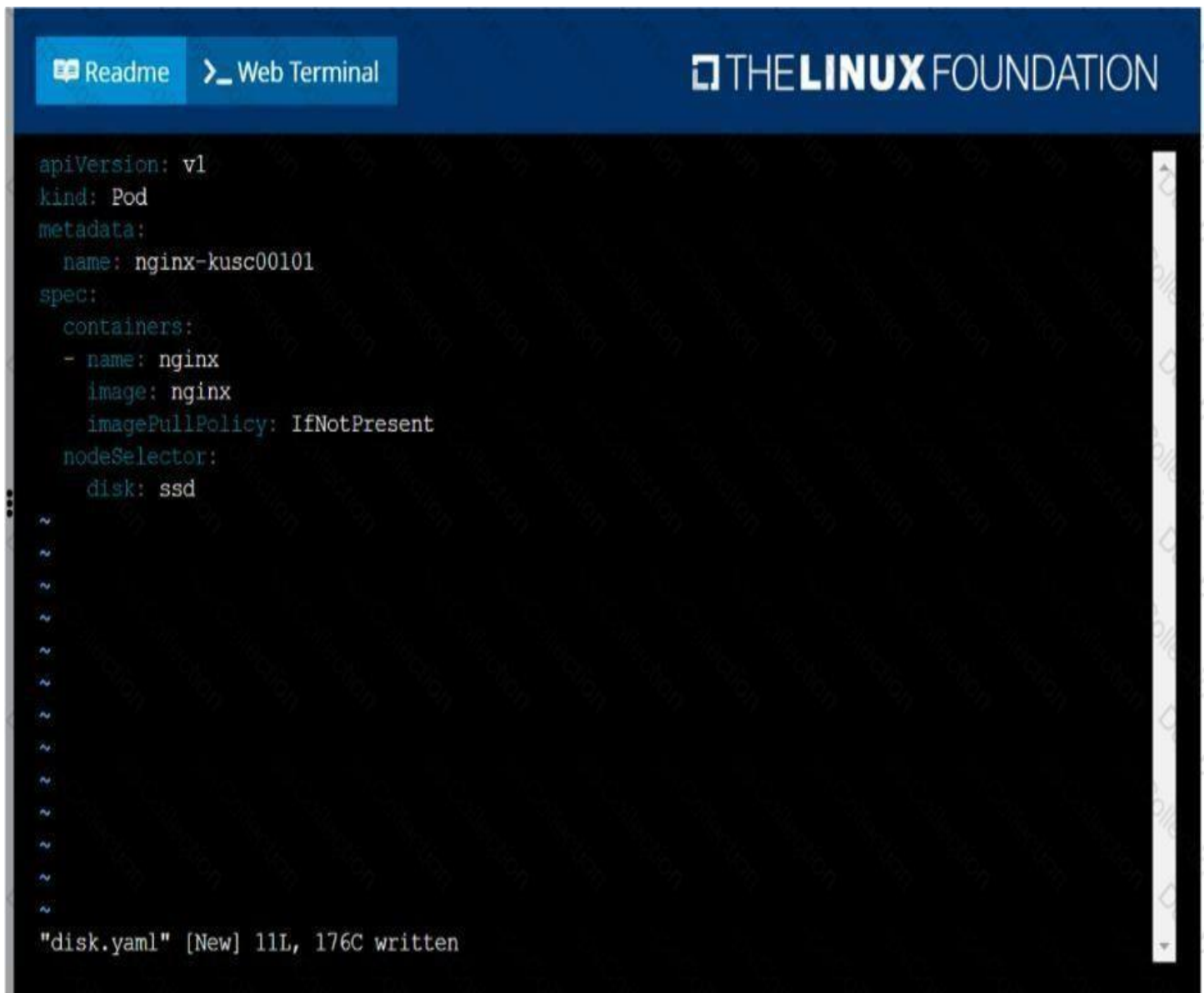
See the solution below.

### Explanation

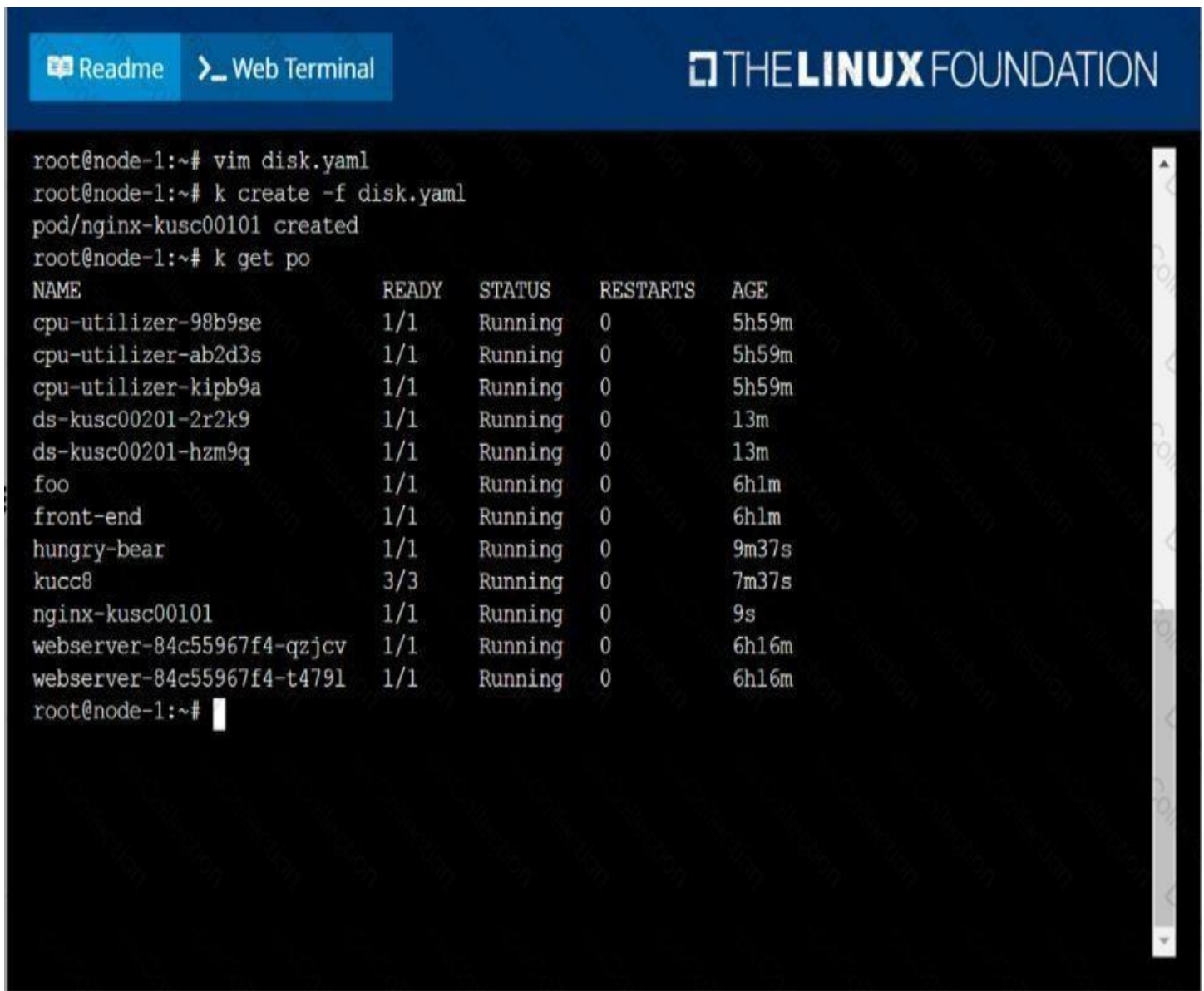
solution

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





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



```
root@node-1:~# vim disk.yaml
root@node-1:~# k create -f disk.yaml
pod/nginx-kusc00101 created
root@node-1:~# k get po
```

NAME	READY	STATUS	RESTARTS	AGE
cpu-utilizer-98b9se	1/1	Running	0	5h59m
cpu-utilizer-ab2d3s	1/1	Running	0	5h59m
cpu-utilizer-kipb9a	1/1	Running	0	5h59m
ds-kusc00201-2r2k9	1/1	Running	0	13m
ds-kusc00201-hzm9q	1/1	Running	0	13m
foo	1/1	Running	0	6h1m
front-end	1/1	Running	0	6h1m
hungry-bear	1/1	Running	0	9m37s
kucc8	3/3	Running	0	7m37s
nginx-kusc00101	1/1	Running	0	9s
webserver-84c55967f4-qzjcv	1/1	Running	0	6h16m
webserver-84c55967f4-t479l	1/1	Running	0	6h16m

```
root@node-1:~#
```

#### Question #:17

List all the pods showing name and namespace with a json path expression

See the solution below.

#### Explanation

```
kubectl get pods -o=jsonpath="{.items[*]['metadata.name',
'metadata.namespace']}"
```

#### Question #:18

A Kubernetes worker node, named `wk8s-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] $ / ssh wk8s-node-0
```

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

```
[student@wk8s-node-0] $ / sudo -i
```

See the solution below.

## Explanation

solution

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



The screenshot shows a web terminal window with a dark blue header. On the left, there are two tabs: 'Readme' and 'Web Terminal'. On the right, the 'THE LINUX FOUNDATION' logo is displayed. The terminal content shows a user at 'root@node-1' running several commands. The first command is 'kubectl config use-context wk8s', which outputs 'Switched to context "wk8s".'. The second command is 'k get nodes', which outputs a table of node information. The table has five columns: NAME, STATUS, ROLES, AGE, and VERSION. It lists three nodes: 'wk8s-master-0' (Ready, master, 77d, v1.18.2), 'wk8s-node-0' (NotReady, <none>, 77d, v1.18.2), and 'wk8s-node-1' (Ready, <none>, 77d, v1.18.2). The third command is 'ssh wk8s-node-0', which has been entered but its output is not visible. A vertical scrollbar is on the right side of the terminal area.

```
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\abc\CKA\20 D.JPG



The screenshot shows a web terminal interface with a dark blue header. On the left, there are two tabs: 'Readme' (with a book icon) and 'Web Terminal' (with a terminal icon). On the right, the 'THE LINUX FOUNDATION' logo is displayed. The terminal window has a black background with white text. It shows a table of node statuses, a successful SSH connection to 'wk8s-node-0', and the Ubuntu 16.04.6 LTS login banner. The banner includes links for documentation, management, and support, as well as information about Kubernetes 1.19 and available updates. Finally, it shows the execution of 'sudo -i' to become root and 'systemctl restart kubelet' and 'systemctl enable kubelet' to manage the Kubernetes service.

```
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\abc\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:~#
```

### Question #:19

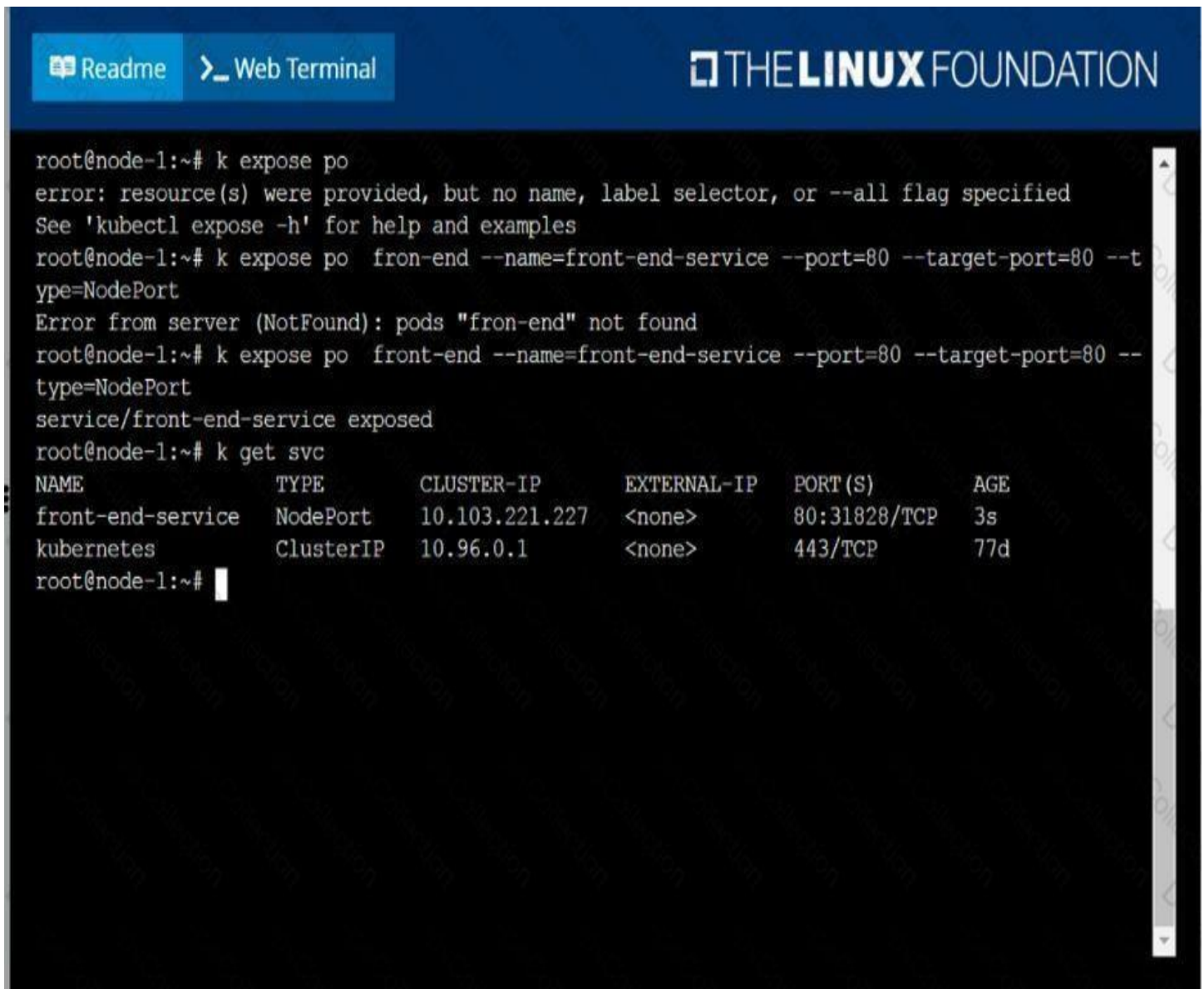
Create and configure the service *front-end-services* so it's accessible through *NodePort* and routes to the existing pod named *front-end*.

See the solution below.

### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\8 B.JPG



```
root@node-1:~# k expose po
error: resource(s) were provided, but no name, label selector, or --all flag specified
See 'kubectl expose -h' for help and examples
root@node-1:~# k expose po fron-end --name=front-end-service --port=80 --target-port=80 --t
ype=NodePort
Error from server (NotFound): pods "fron-end" not found
root@node-1:~# k expose po front-end --name=front-end-service --port=80 --target-port=80 --
type=NodePort
service/front-end-service exposed
root@node-1:~# k get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
front-end-service	NodePort	10.103.221.227	<none>	80:31828/TCP	3s
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	77d

```
root@node-1:~#
```

### Question #:20

Create an nginx pod and list the pod with different levels of verbosity

See the solution below.

### Explanation

// create a pod

```
kubectl run nginx --image=nginx --restart=Never --port=80
```

// List the pod with different verbosity

```
kubectl get po nginx --v=7
```

```
kubectl get po nginx --v=8
```

```
kubectl get po nginx --v=9
```

### Question #:21

List all the pods sorted by created timestamp

See the solution below.

### Explanation

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

### Question #:22

Perform the following tasks:

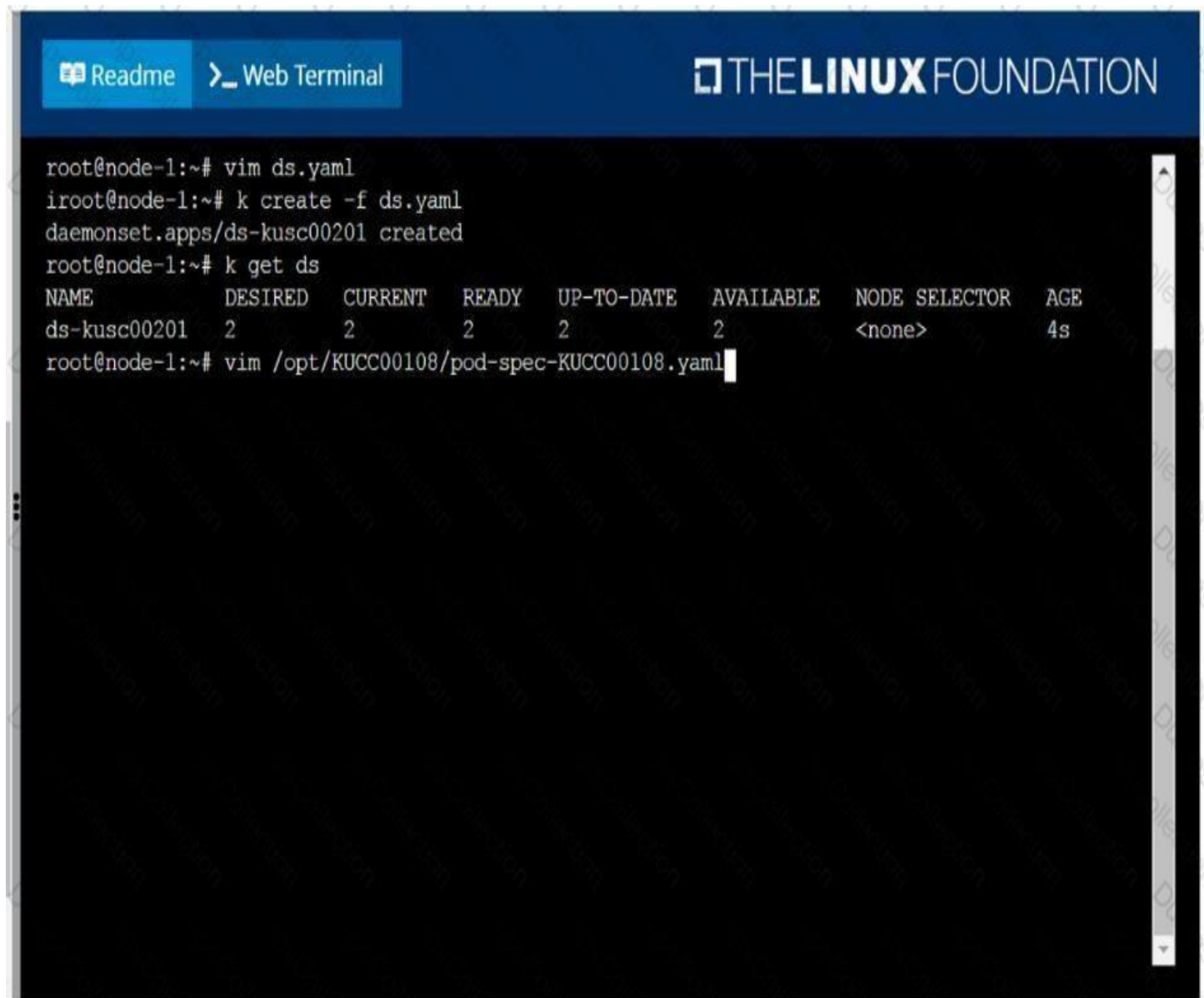
- Add an init container tohungry-bear(which has beendefined in spec file /opt/KUCC00108/pod-spec-KUCC00108.yaml)
- The init container should createan empty file named/workdir/calm.txt
- If/workdir/calm.txtis notdetected, the pod should exit
- Once the spec file has beenupdatedwith the init containerdefinition, the pod should becreated

See the solution below.

### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\4 B.JPG



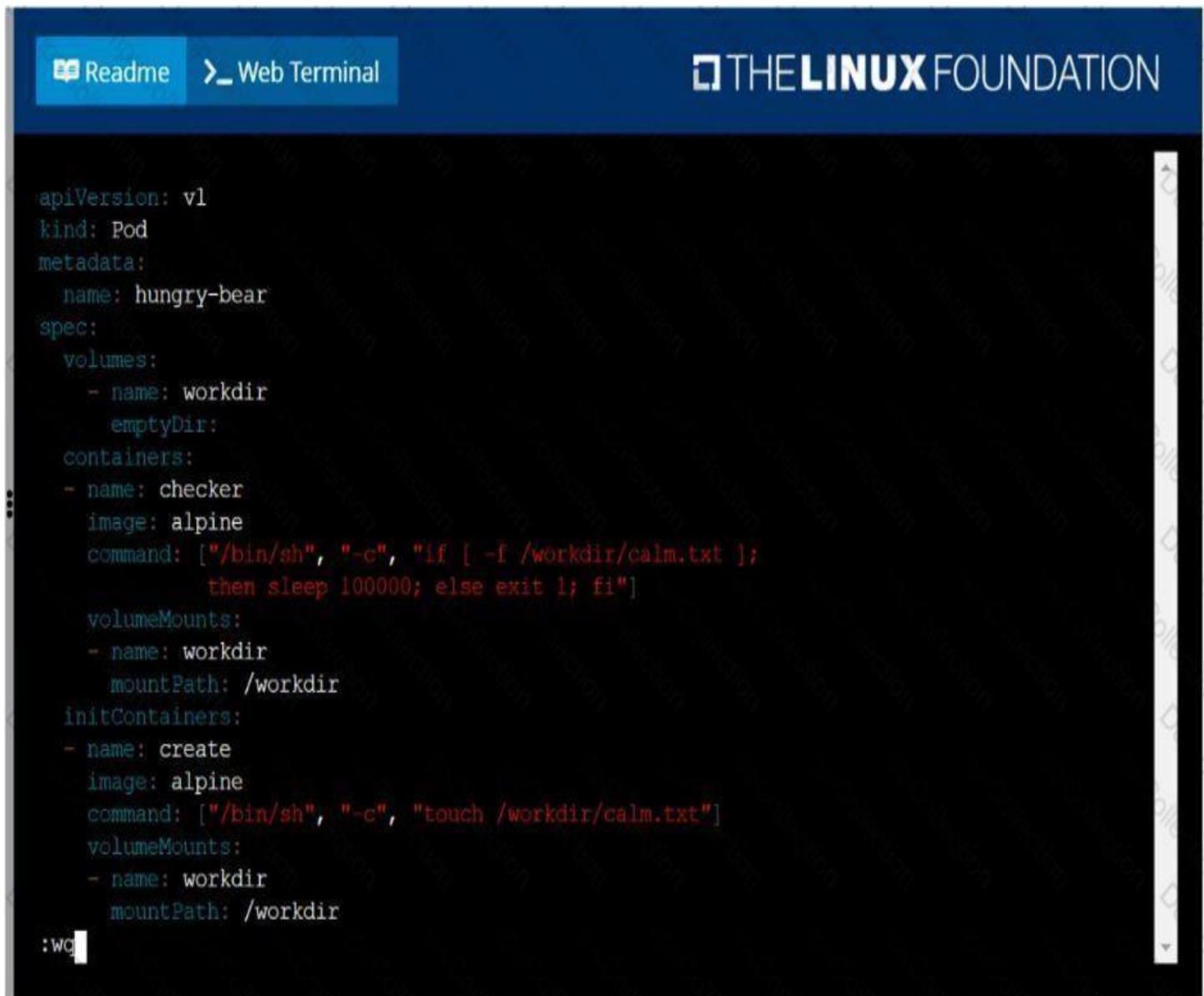
The screenshot shows a web terminal window with a dark blue header. On the left, there are two tabs: 'Readme' and 'Web Terminal'. On the right, the 'THE LINUX FOUNDATION' logo is displayed. The terminal content shows a series of commands and their outputs:

```
root@node-1:~# vim ds.yaml
iroot@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
```

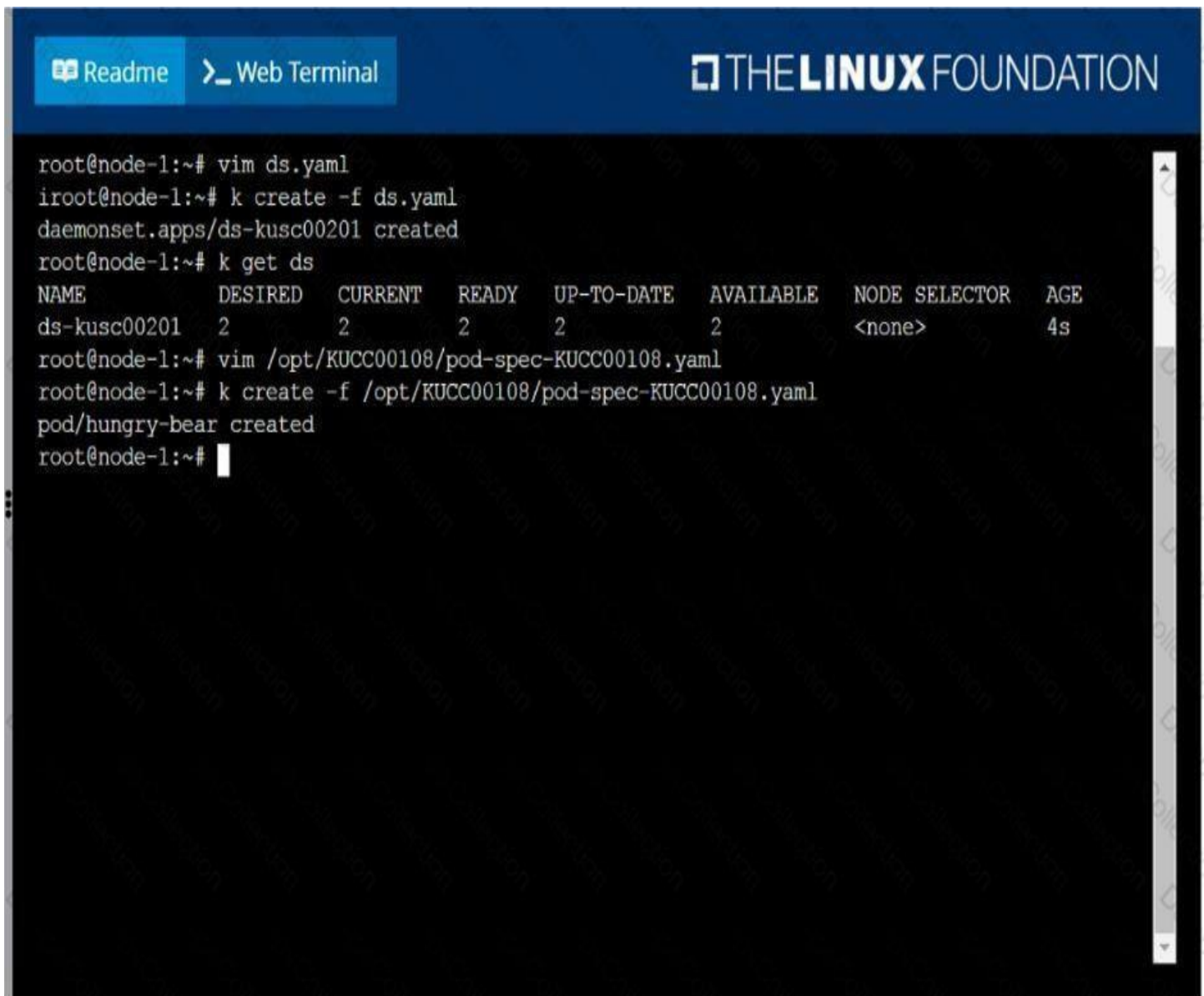
F:\Work\Data Entry Work\Data Entry\abc\CKA\4 C.JPG



The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with a 'Readme' button and a 'Web Terminal' tab. The 'THE LINUX FOUNDATION' logo is in the top right corner. The terminal displays a Kubernetes manifest for a pod named 'hungry-bear'. The manifest includes a 'spec' section with 'volumes' and 'containers'. The 'checker' container runs an Alpine image with a command that checks for the existence of '/workdir/calm.txt' and sleeps for 100,000 seconds if it exists. The 'initContainers' section includes a 'create' container that runs an Alpine image with a command to create the file '/workdir/calm.txt'. The terminal ends with a prompt ':wq'.

```
apiVersion: v1
kind: Pod
metadata:
  name: hungry-bear
spec:
  volumes:
    - name: workdir
      emptyDir: {}
  containers:
    - name: checker
      image: alpine
      command: ["/bin/sh", "-c", "if [ -f /workdir/calm.txt ];
        then sleep 100000; else exit 1; fi"]
      volumeMounts:
        - name: workdir
          mountPath: /workdir
  initContainers:
    - name: create
      image: alpine
      command: ["/bin/sh", "-c", "touch /workdir/calm.txt"]
      volumeMounts:
        - name: workdir
          mountPath: /workdir
:wq
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\4 D.JPG



```
root@node-1:~# vim ds.yaml
iroot@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:~#
```

### Question #:23

Check the image version in pod without the describe command

See the solution below.

### Explanation

kubectrl get po nginx -o

jsonpath='{.spec.containers[].image} {"\n"}'

### Question #:24



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`.

See the solution below.

## Explanation

solution

### Persistent Volume

A persistent volume is a piece of storage in a Kubernetes cluster. Persistent Volumes 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 Persistent Volume provisioned in an easy way.

### Creating Persistent Volume

```
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`.

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

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 persistent volume to a persistent volume claim.

### Persistent Volume Claim

In a real ecosystem, a system admin will create the Persistent Volume then a developer will create a Persistent Volume Claim which will be referenced in a pod. A Persistent Volume Claim is created by specifying the minimum size and the access mode they require from the persistent volume.

### 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 persistent volume 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: v1
kind: Pod
metadata:
  creationTimestamp: null
name: app-data
spec:
  volumes:
  - name: configpvc
    persistentVolumeClaim:
      claimName: app-data
  containers:
  - image: nginx
    name: configpvc
    volumeMounts:
    - mountPath: "/srv/app-data"
      name: configpvc
```

### Question #:25

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

See the solution below.

### Explanation

```
kubectl get po -o=custom-columns="POD_NAME:.metadata.name,
POD_STATUS:.status.containerStatuses[].state"
```

### Question #:26

Create a pod with environment variables as var1=value1. Check the environment variable in pod

See the solution below.

### Explanation

```
kubectl run nginx --image=nginx --restart=Never --env=var1=value1
```

# then

```
kubectl exec -it nginx -- env
```

# or

```
kubectl exec -it nginx -- sh -c 'echo $var1'
```

# or

```
kubectl describe po nginx | grep value1
```

### Question #:27

For this item, you will have to *ssh* to the nodes *ik8s-master-0* and *ik8s-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=alloptions*.

- Configure the node *ik8s-master-0* as a master node. .
- Join the node *ik8s-node-0* to the cluster.

See the solution below.

### 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.

### Question #:28

**Create a namespace called 'development' and a pod with image *nginx* called *nginx* on this namespace.**

See the solution below.

### Explanation

```
kubectl create namespace development
```

```
kubectl run nginx --image=nginx --restart=Never -n development
```

### Question #:29

List all the pods sorted by name

See the solution below.

### Explanation

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

#### Question #:30

List “nginx-dev” and “nginx-prod” pod and delete those pods

See the solution below.

### Explanation

```
kubect1 get pods -o wide
```

```
kubectl delete po “nginx-dev”kubectl delete po “nginx-prod”
```

#### Question #:31

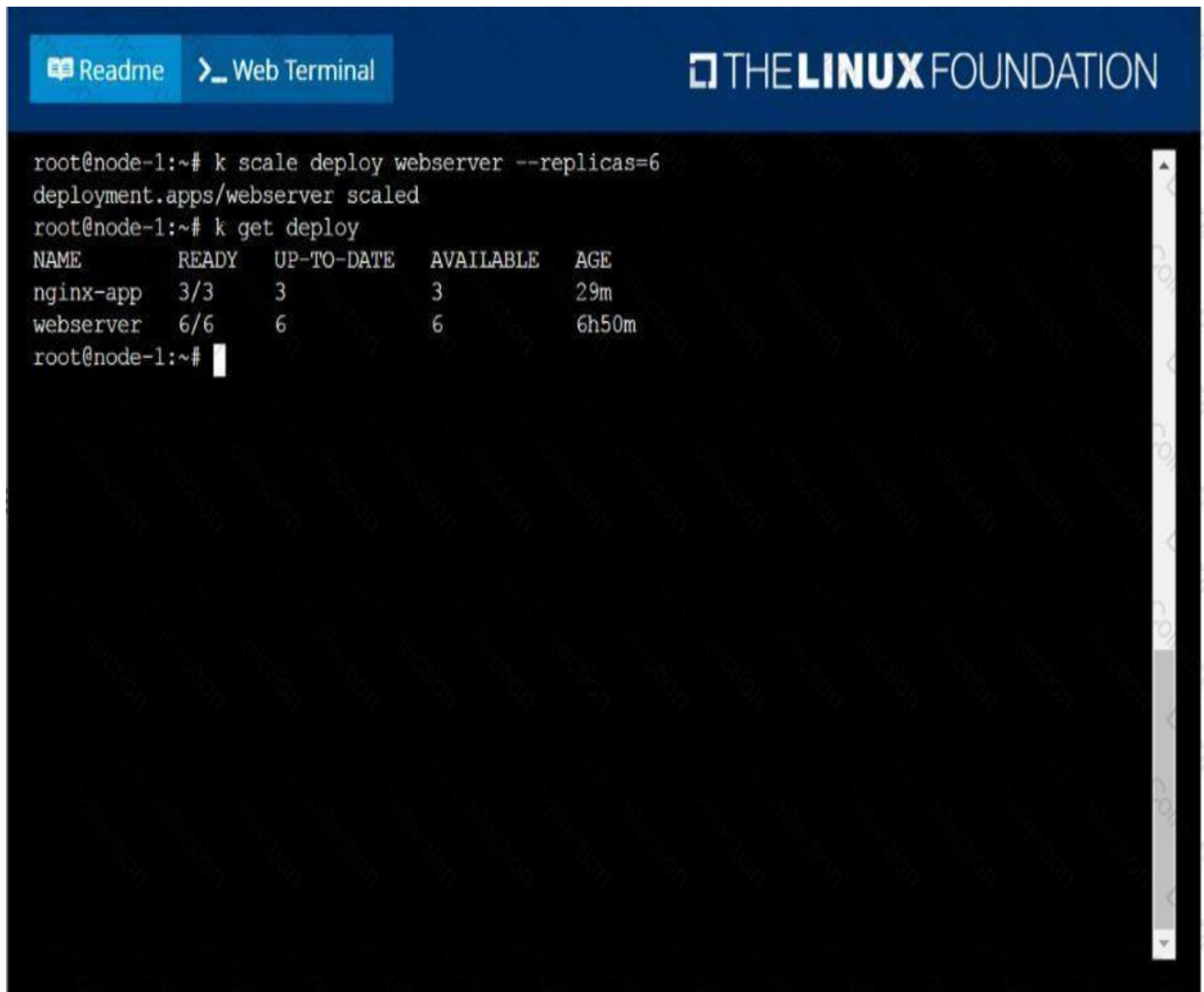
Scale the deployment*webserver*to6pods.

See the solution below.

### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\14 B.JPG



The screenshot shows a terminal window titled "Web Terminal" with the Linux Foundation logo. The user is at the root of a node-1 and runs the command `k scale deploy webserver --replicas=6`. The output shows the deployment `deployment.apps/webserver` being scaled. Then, the user runs `k get deploy`, which displays a table of deployments.

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx-app	3/3	3	3	29m
webserver	6/6	6	6	6h50m

The terminal prompt is `root@node-1:~#`.

### Question #:32

**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.**

See the solution below.

### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\2 C.JPG

```

77d
pv0007 7Gi RWO Recycle Available slow
77d
pv0006 8Gi RWO Recycle Available slow
77d
pv0003 10Gi RWO Recycle Available slow
77d
pv0002 11Gi RWO Recycle Available slow
77d
pv0010 13Gi RWO Recycle Available slow
77d
pv0011 14Gi RWO Recycle Available slow
77d
pv0001 16Gi RWO Recycle Available slow
77d
pv0009 17Gi RWO Recycle Available slow
77d
pv0005 18Gi RWO Recycle Available slow
77d
pv0008 19Gi RWO Recycle Available slow
77d
pv0000 21Gi RWO Recycle Available slow
77d
root@node-1:~# k get pv --sort-by=.spec.capacity.storage > /opt/KUCC00102/volume_list
root@node-1:~#

```

### Question #:33

Create a pod named **kucc8** with a single app container for each of the following images running inside (there may be between 1 and 4 images specified):

**nginx + redis + memcached.**

See the solution below.

### Explanation

solution

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





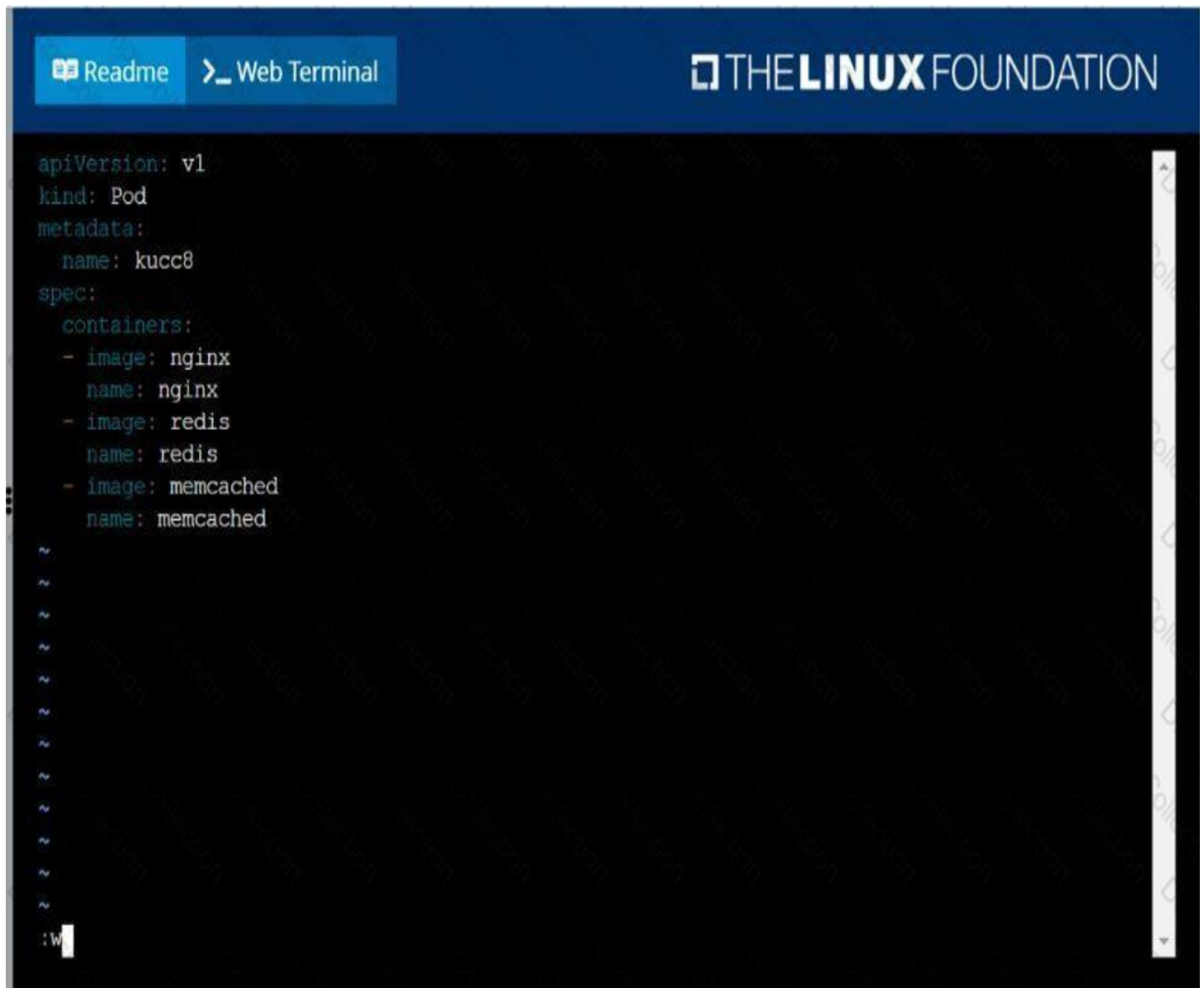
The screenshot shows a terminal window with a dark background and light text. At the top, there is a blue header bar with a 'Readme' button and a 'Web Terminal' link. The 'THE LINUX FOUNDATION' logo is on the right. The terminal content shows a series of commands and their outputs:

```

root@node-1:~# vim ds.yaml
iroot@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\abc\CKA\5 C.JPG



The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with two buttons: 'Readme' and 'Web Terminal'. To the right of the buttons is the 'THE LINUX FOUNDATION' logo. The main area of the terminal displays a YAML manifest for a Pod. The manifest includes the following fields: 'apiVersion: v1', 'kind: Pod', 'metadata: name: kucc8', and 'spec: containers:'. Under 'containers', there are three entries: 'nginx', 'redis', and 'memcached', each with 'image' and 'name' fields. The terminal also shows a vertical scrollbar on the right and a prompt character '~' at the bottom left.

```
apiVersion: v1
kind: Pod
metadata:
  name: kucc8
spec:
  containers:
  - image: nginx
    name: nginx
  - image: redis
    name: redis
  - image: memcached
    name: memcached
```

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



```

cpu-utilizer-98b9se      1/1      Running      0          5h51m
cpu-utilizer-ab2d3s      1/1      Running      0          5h51m
cpu-utilizer-kipb9a      1/1      Running      0          5h51m
ds-kusc00201-2r2k9       1/1      Running      0          6m12s
ds-kusc00201-hzm9q       1/1      Running      0          6m12s
foo                      1/1      Running      0          5h54m
front-end                1/1      Running      0          5h53m
hungry-bear              1/1      Running      0          2m4s
kucc8                    0/3      ContainerCreating 0          4s
webserver-84c55967f4-qzjcv 1/1      Running      0          6h9m
webserver-84c55967f4-t4791 1/1      Running      0          6h9m
root@node-1:~# k get po
NAME                                READY   STATUS    RESTARTS   AGE
cpu-utilizer-98b9se                 1/1     Running   0          5h52m
cpu-utilizer-ab2d3s                 1/1     Running   0          5h52m
cpu-utilizer-kipb9a                 1/1     Running   0          5h52m
ds-kusc00201-2r2k9                  1/1     Running   0          6m31s
ds-kusc00201-hzm9q                  1/1     Running   0          6m31s
foo                                 1/1     Running   0          5h54m
front-end                           1/1     Running   0          5h54m
hungry-bear                         1/1     Running   0          2m23s
kucc8                               3/3     Running   0          23s
webserver-84c55967f4-qzjcv          1/1     Running   0          6h9m
webserver-84c55967f4-t4791          1/1     Running   0          6h9m
root@node-1:~#

```

**Question #34**

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

See the solution below.

**Explanation**

```
kubectl get po --all-namespaces > /opt/pods-list.yaml
```

**Question #35**

Ensure a single instance of podnginxis running on each node of theKubernetes cluster wherenginxalso represents the Image name whichhas to be used. Do not override anytaints currently in place.

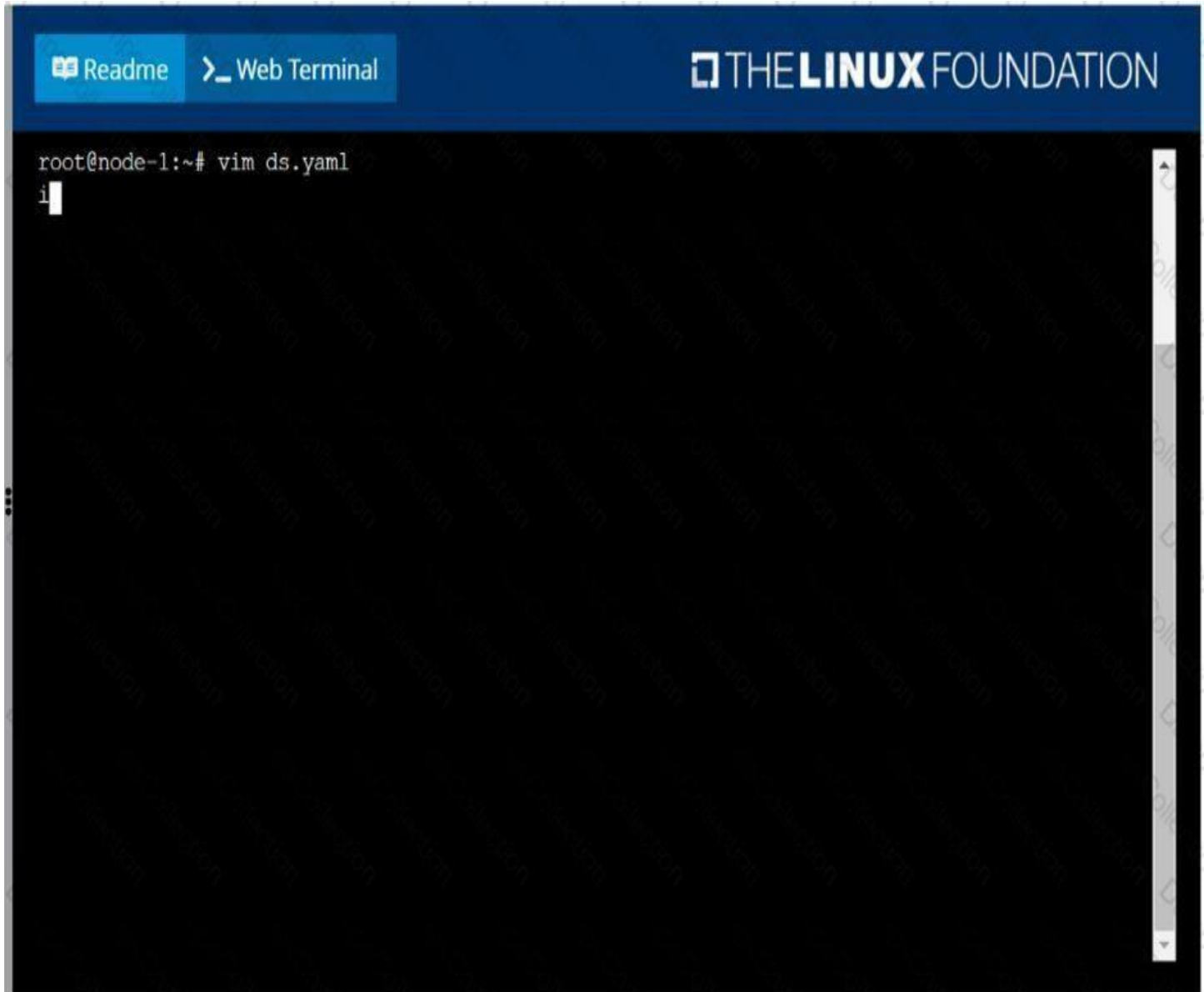
Use **DaemonSet** to complete this task and use *ds-kusc00201* as DaemonSet name.

See the solution below.

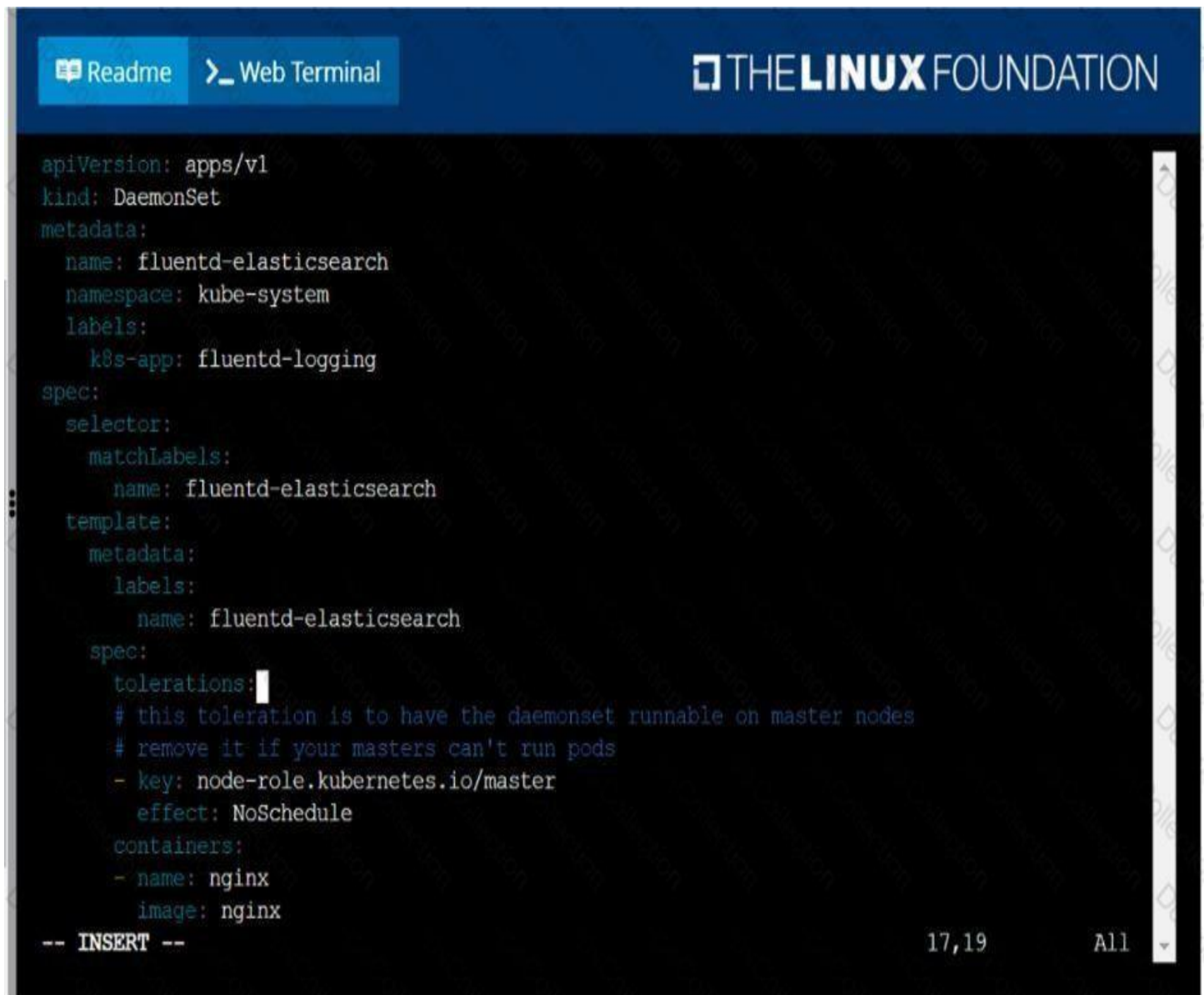
## Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\3 B.JPG




F:\Work\Data Entry Work\Data Entry\abc\CKA\3 C.JPG



The screenshot shows a web terminal interface with a dark background. At the top, there are two tabs: 'Readme' and 'Web Terminal'. The 'Web Terminal' tab is active. The terminal displays a YAML configuration for a Kubernetes DaemonSet. The configuration includes metadata (name, namespace, labels), a selector (matchLabels), and a template (metadata, labels, spec). The spec includes tolerations (to run on master nodes) and containers (nginx). The terminal also shows a cursor at the end of the last line of the spec, and a status bar at the bottom indicating '17,19' and 'All'.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
        # this toleration is to have the daemonset runnable on master nodes
        # remove it if your masters can't run pods
        - key: node-role.kubernetes.io/master
          effect: NoSchedule
      containers:
        - name: nginx
          image: nginx
-- INSERT --
```

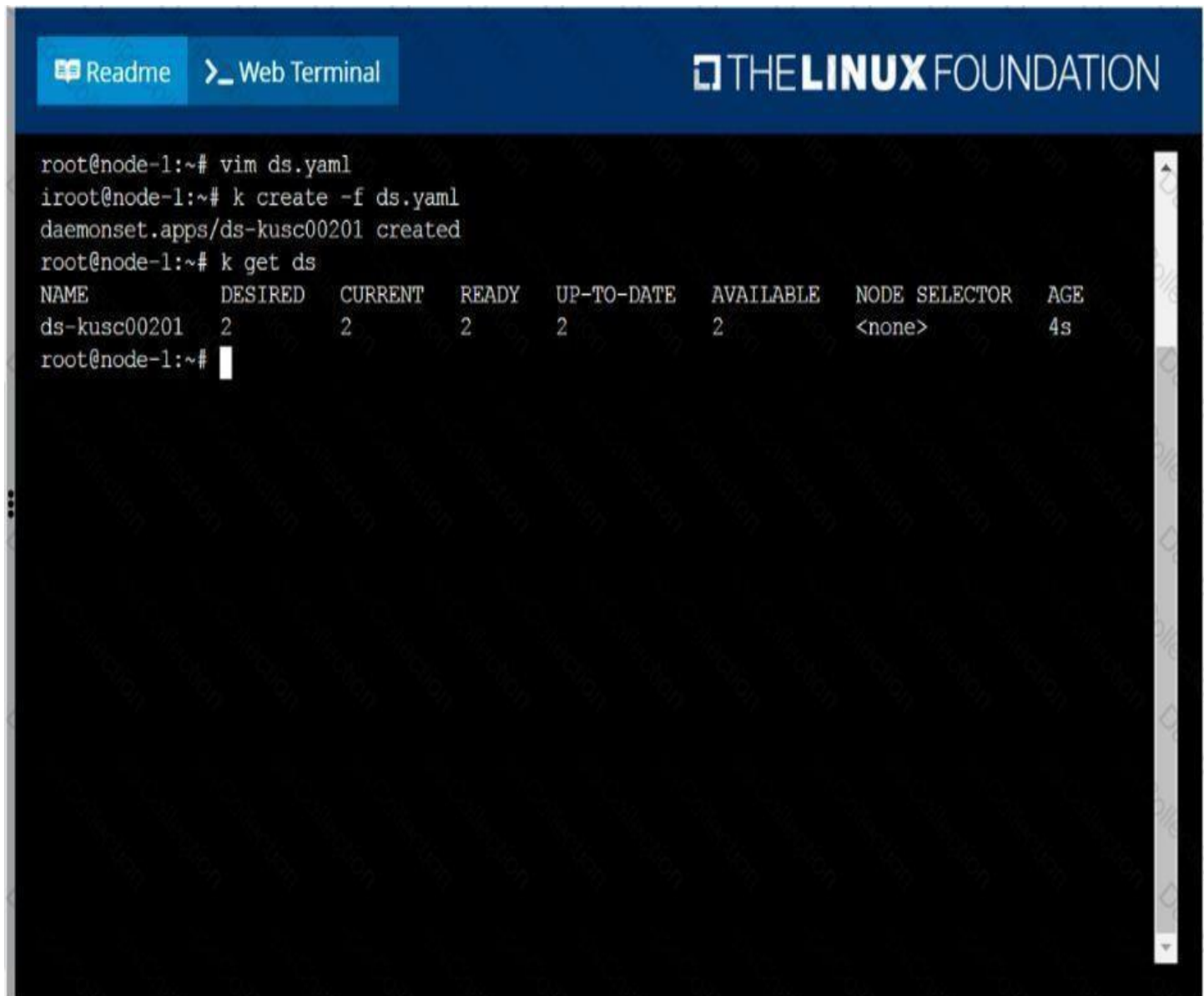
F:\Work\Data Entry Work\Data Entry\abc\CKA\3 D.JPG



The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with a "Readme" button and a "Web Terminal" button. The "THE LINUX FOUNDATION" logo is on the right. The terminal displays a YAML configuration for a DaemonSet. The configuration includes a selector for pods with the label "fluentd-elasticsearch" and a template that defines a container named "nginx" using the "nginx" image. The terminal also shows a list of pod names and a prompt for a user named "wg".

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: ds-kusc00201
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: nginx
        image: nginx
~
~
~
~
~
~
~
~
:wg
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\3 E.JPG



```
root@node-1:~# vim ds.yaml
iroot@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:~#
```

### Question #36

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

See the solution below.

### Explanation

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

### Question #37

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

See the solution below.

## 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

```
kubectrl 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

#### Question #:38

Get list of all the pods showing name and namespace with a jsonpath expression.

See the solution below.

#### Explanation

```
kubectl get pods -o=jsonpath="{.items[*]['metadata.name']  
, 'metadata.namespace']}"
```

#### Question #:39

Create a pod that echo “hello world” and then exists. Have the pod deleted automatically when it’s completed

See the solution below.

#### Explanation

```
kubectl run busybox --image=busybox -it --rm --restart=Never --  
/bin/sh -c 'echo hello world'  
  
kubectl get po # You shouldn't see pod with the name "busybox"
```

#### Question #:40

Create a busybox pod that runs the command “env” and save the output to “envpod” file

See the solution below.

#### Explanation

```
kubectl run busybox --image=busybox --restart=Never --rm -it -- env > envpod.yaml
```

#### Question #:41

Create a file:

```
/opt/KUCC00302/kucc00302.txt that lists all pods that implement service baz in namespace development.
```



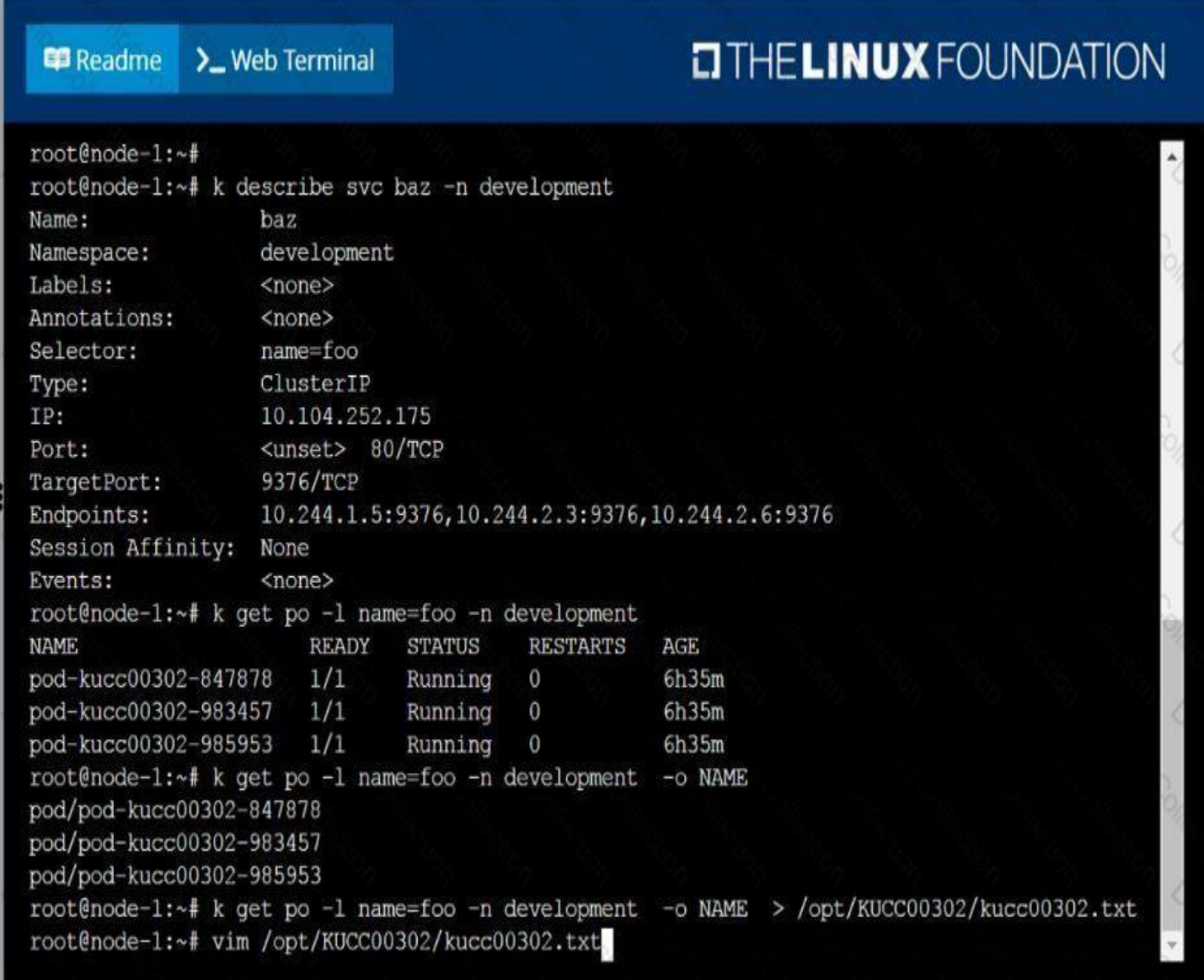
The format of the file should be onepod name per line.

See the solution below.

## Explanation

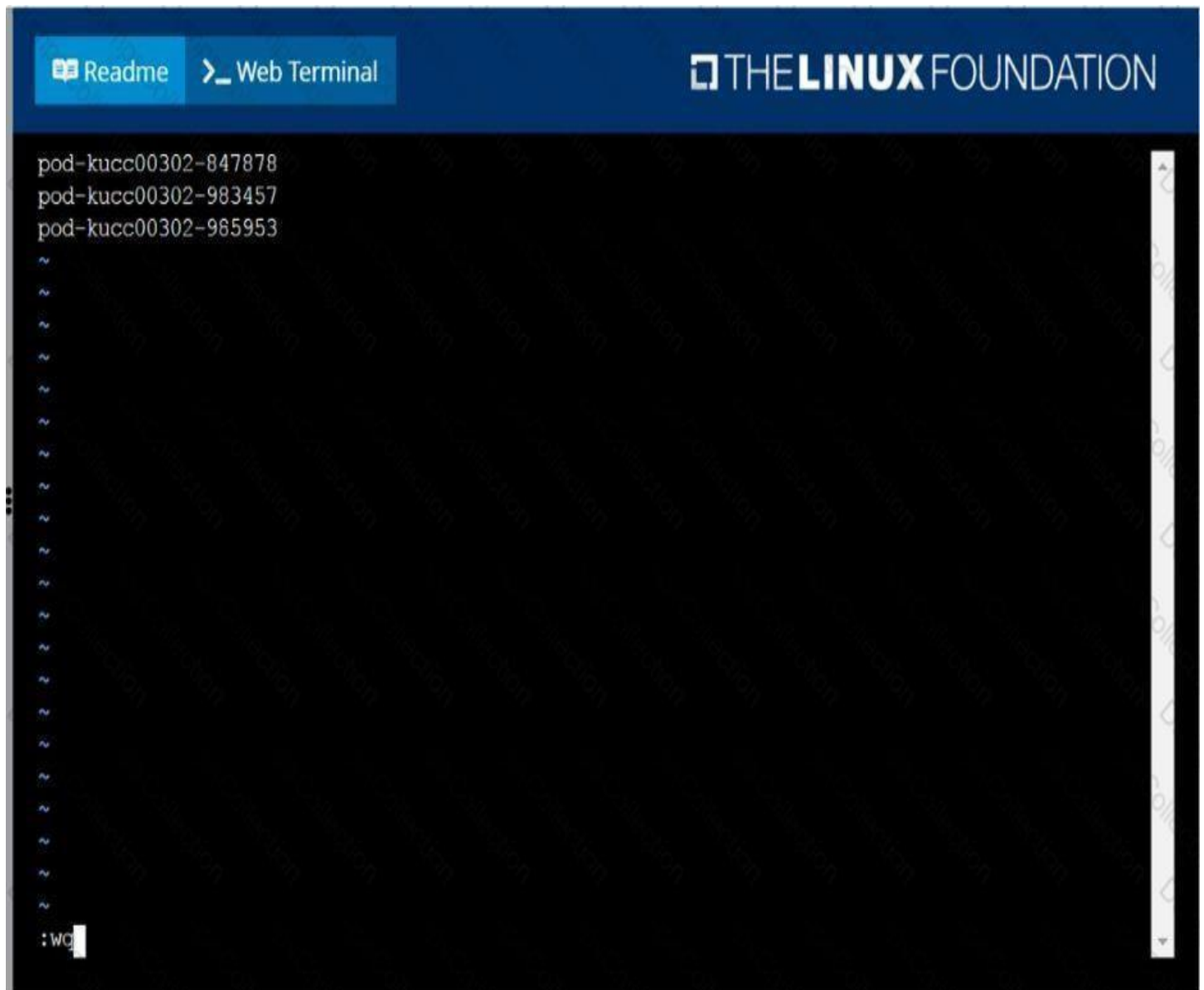
solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\11 B.JPG



```
root@node-1:~#
root@node-1:~# k describe svc baz -n development
Name:          baz
Namespace:     development
Labels:        <none>
Annotations:   <none>
Selector:      name=foo
Type:          ClusterIP
IP:            10.104.252.175
Port:          <unset> 80/TCP
TargetPort:    9376/TCP
Endpoints:     10.244.1.5:9376,10.244.2.3:9376,10.244.2.6:9376
Session Affinity: None
Events:        <none>
root@node-1:~# k get po -l name=foo -n development
NAME                READY   STATUS    RESTARTS   AGE
pod-kucc00302-847878 1/1     Running   0           6h35m
pod-kucc00302-983457 1/1     Running   0           6h35m
pod-kucc00302-985953 1/1     Running   0           6h35m
root@node-1:~# k get po -l name=foo -n development -o NAME
pod/pod-kucc00302-847878
pod/pod-kucc00302-983457
pod/pod-kucc00302-985953
root@node-1:~# k get po -l name=foo -n development -o NAME > /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\11 C.JPG



F:\Work\Data Entry Work\Data Entry\abc\CKA\11 D.JPG

```

Name:          baz
Namespace:     development
Labels:        <none>
Annotations:   <none>
Selector:      name=foo
Type:          ClusterIP
IP:            10.104.252.175
Port:          <unset> 80/TCP
TargetPort:    9376/TCP
Endpoints:     10.244.1.5:9376,10.244.2.3:9376,10.244.2.6:9376
Session Affinity: None
Events:        <none>
root@node-1:~# k get po -l name=foo -n development
NAME                                READY   STATUS    RESTARTS   AGE
pod-kucc00302-847878                1/1     Running   0           6h35m
pod-kucc00302-983457                1/1     Running   0           6h35m
pod-kucc00302-985953                1/1     Running   0           6h35m
root@node-1:~# k get po -l name=foo -n development -o NAME
pod/pod-kucc00302-847878
pod/pod-kucc00302-983457
pod/pod-kucc00302-985953
root@node-1:~# k get po -l name=foo -n development -o NAME > /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
root@node-1:~# vim /opt/KUCC00302/kucc00302.txt
root@node-1:~#

```

### Question #:42


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

See the solution below.

### Explanation

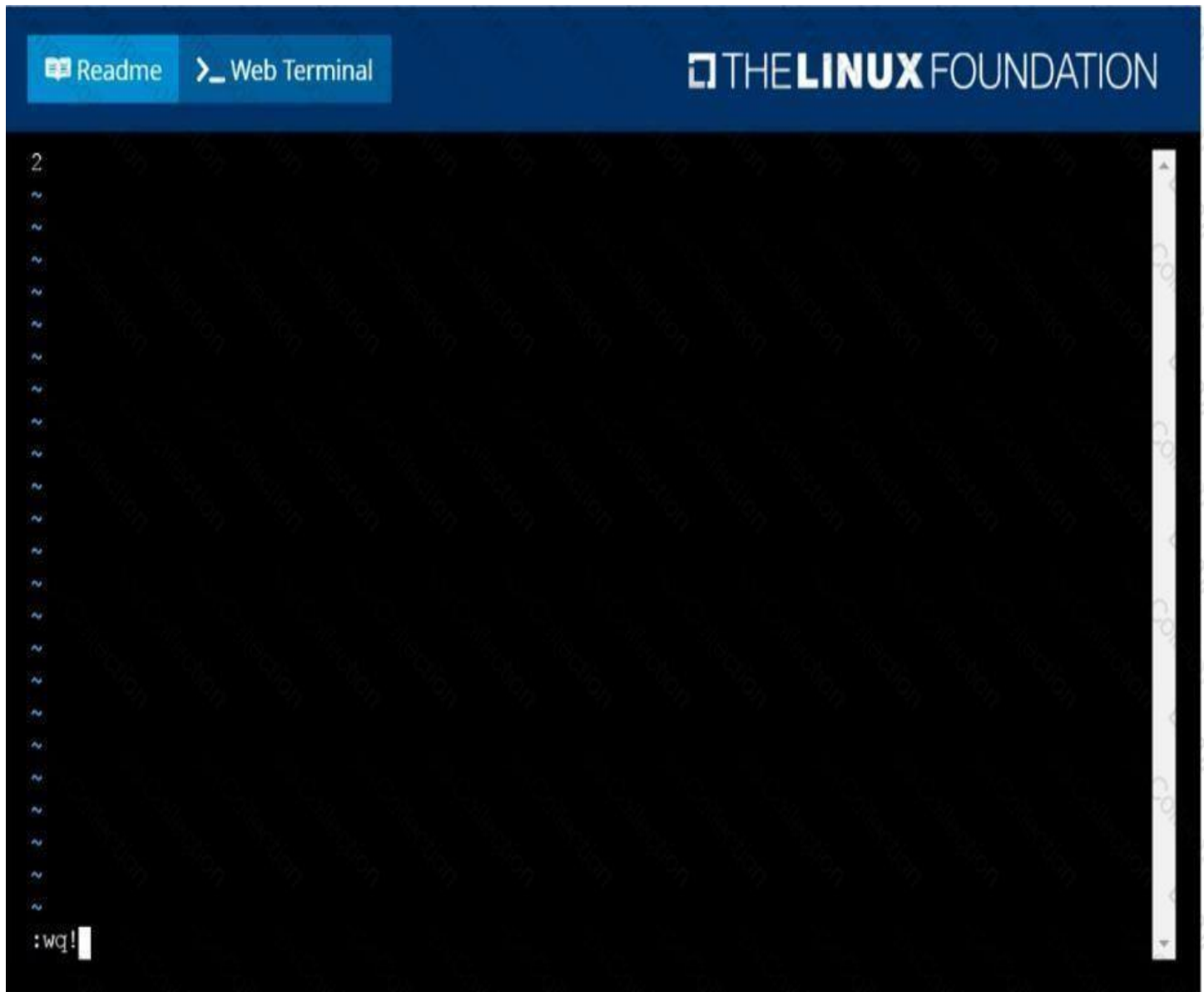
solution

*F:\Work\Data Entry Work\Data Entry\abc\CKA\15 B.JPG*



```
root@node-1:~# k scale deploy webserver --replicas=6
deployment.apps/webserver scaled
root@node-1:~# k get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
nginx-app     3/3     3            3           29m
webserver     6/6     6            6           6h50m
root@node-1:~#
root@node-1:~# k get nodes
NAME           STATUS   ROLES    AGE   VERSION
k8s-master-0   Ready   master   77d   v1.18.2
k8s-node-0     Ready   <none>   77d   v1.18.2
k8s-node-1     Ready   <none>   77d   v1.18.2
root@node-1:~# vim /opt/KUCC00104/kucc00104.txt
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\15 C.JPG



#### Question #:43

Create a busybox pod and add "sleep 3600" command

See the solution below.

#### Explanation

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

```
"sleep 3600"
```

#### Question #:44

Configure the kubelet systemd-managed service, on the node labelled with `name=wk8s-node-1`, to launch a pod containing a single container of Image `httpd` named `webtool` automatically. Any spec files required should be placed in the `/etc/kubernetes/manifests` directory on the node.

You can ssh to the appropriate node using:

```
[student@node-1] $ ssh wk8s-node-1
```

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

```
[student@wk8s-node-1] $ /sudo -i
```

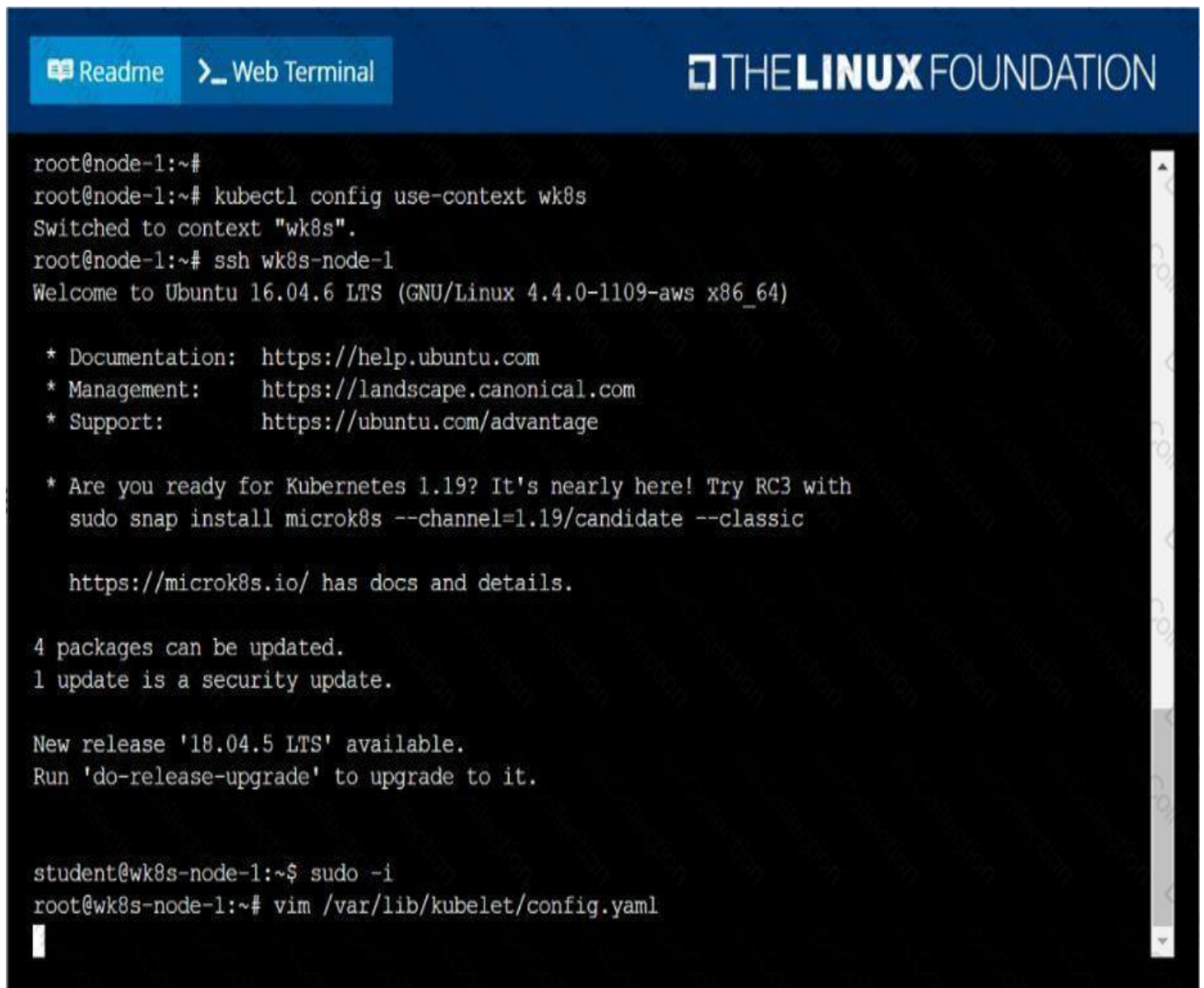
See the solution below.

## Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\21 C.JPG





```
Readme Web Terminal THE LINUX FOUNDATION

root@node-1:~#
root@node-1:~# kubectl config use-context wk8s
Switched to context "wk8s".
root@node-1:~# ssh wk8s-node-1
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.

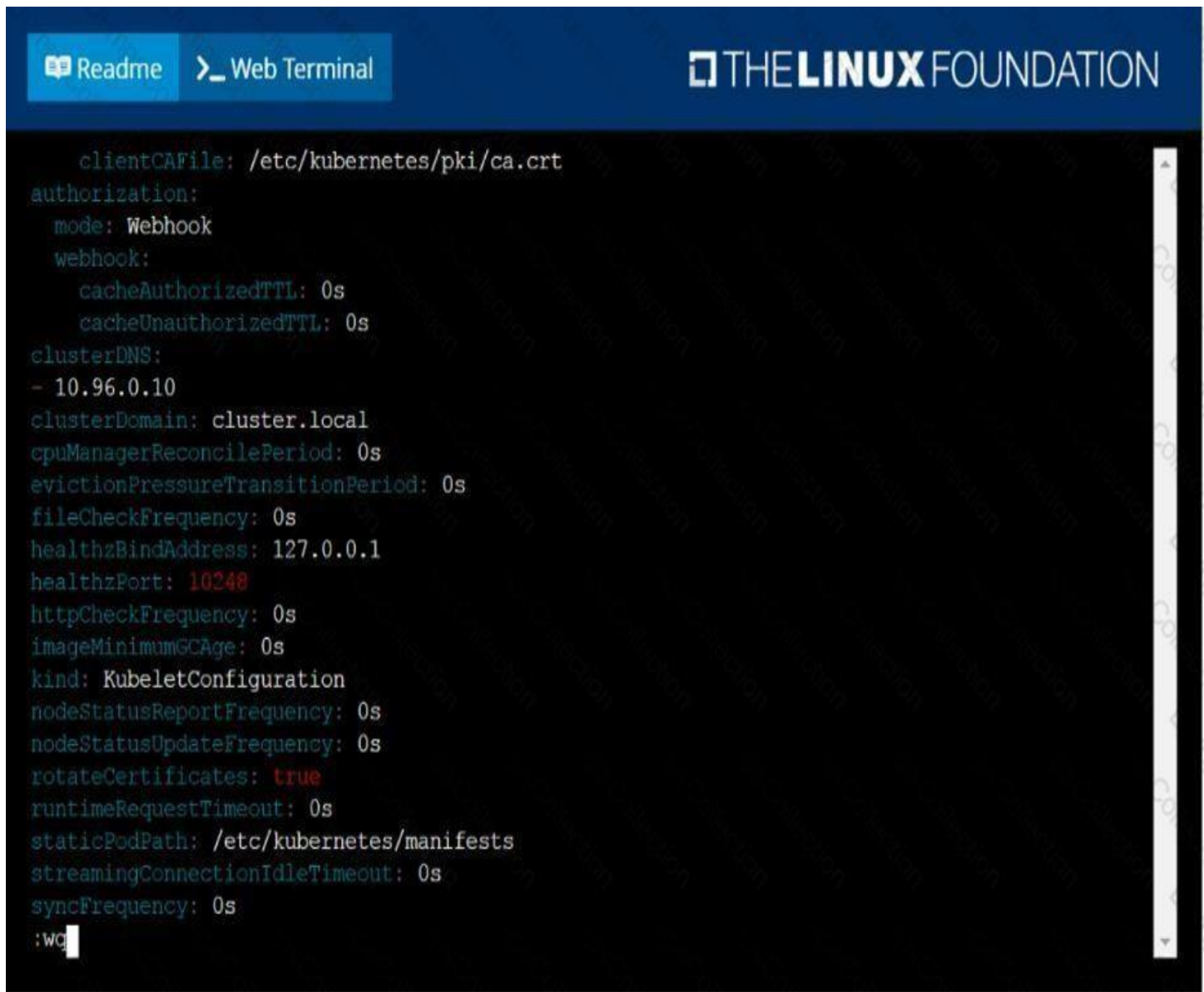
4 packages can be updated.
1 update is a security update.

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

student@wk8s-node-1:~$ sudo -i
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml
█
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\21 D.JPG





```
clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
:wc
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\21 E.JPG



```
root@node-1:~# ssh wk8s-node-1
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

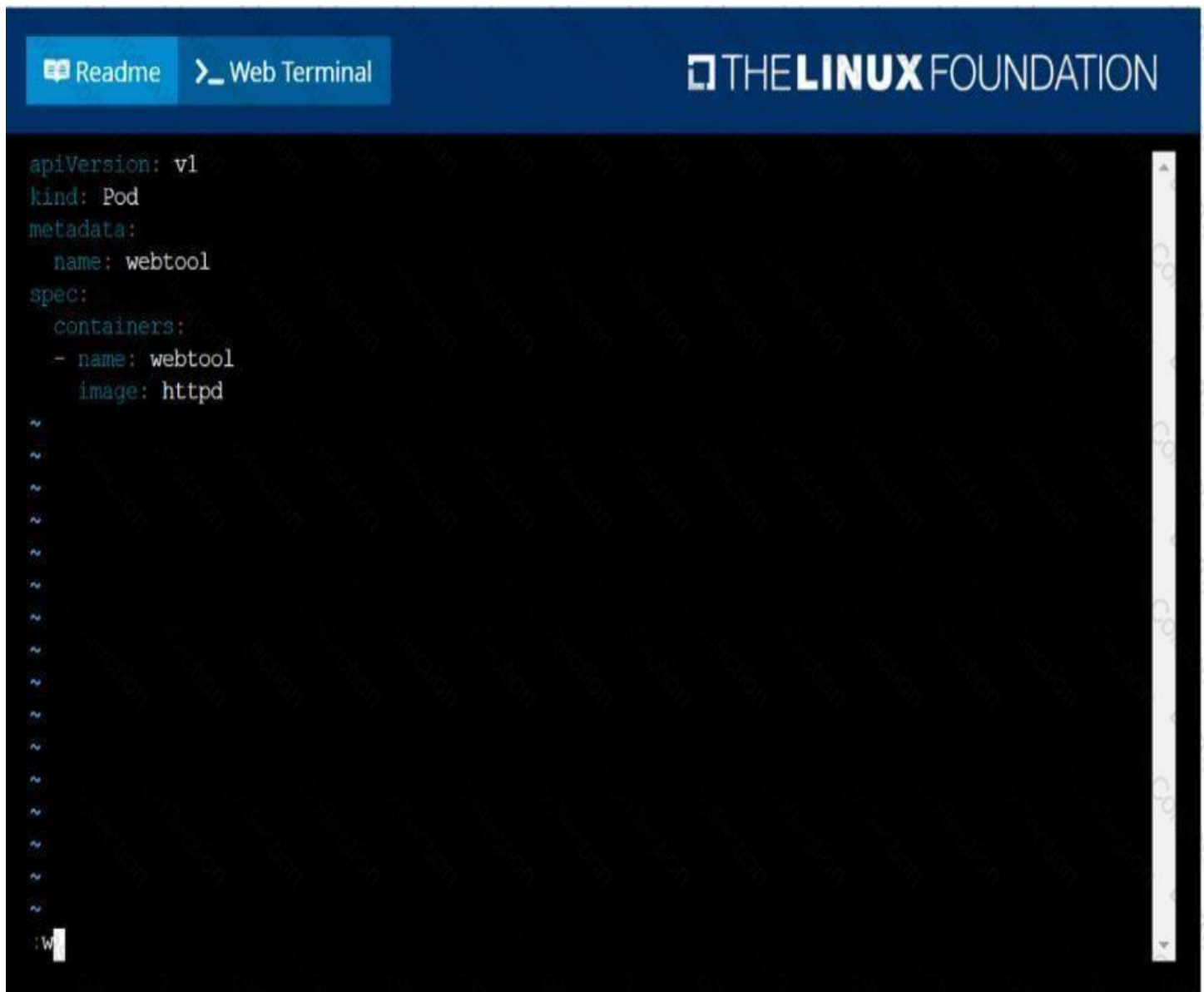
   https://microk8s.io/ has docs and details.

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-1:~$ sudo -i
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml
root@wk8s-node-1:~# cd /etc/kubernetes/manifests
root@wk8s-node-1:/etc/kubernetes/manifests#
root@wk8s-node-1:/etc/kubernetes/manifests# vim pod.yaml
```

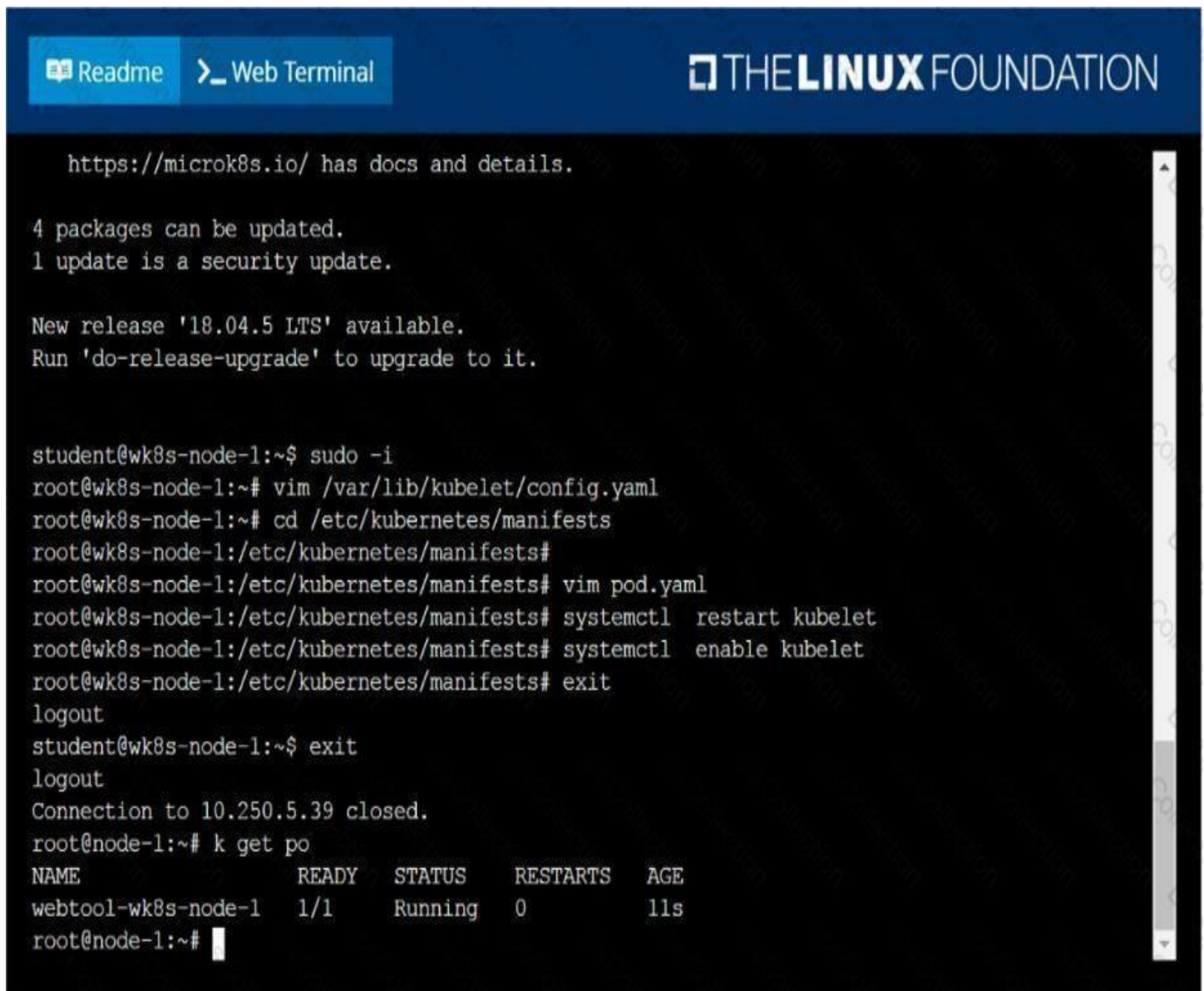
F:\Work\Data Entry Work\Data Entry\abc\CKA\21 F.JPG



The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with two tabs: 'Readme' and 'Web Terminal'. The 'Web Terminal' tab is active. To the right of the tabs is the 'THE LINUX FOUNDATION' logo. The main area of the terminal displays a Kubernetes manifest in JSON format. The manifest defines a Pod named 'webtool' using the 'httpd' image. Below the manifest, there are several tilde (~) characters representing a shell prompt, and a cursor is visible at the bottom left.

```
apiVersion: v1
kind: Pod
metadata:
  name: webtool
spec:
  containers:
  - name: webtool
    image: httpd
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\21 G.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-1:~$ sudo -i
root@wk8s-node-1:~# vim /var/lib/kubelet/config.yaml
root@wk8s-node-1:~# cd /etc/kubernetes/manifests
root@wk8s-node-1:/etc/kubernetes/manifests#
root@wk8s-node-1:/etc/kubernetes/manifests# vim pod.yaml
root@wk8s-node-1:/etc/kubernetes/manifests# systemctl restart kubelet
root@wk8s-node-1:/etc/kubernetes/manifests# systemctl enable kubelet
root@wk8s-node-1:/etc/kubernetes/manifests# exit
logout
student@wk8s-node-1:~$ exit
logout
Connection to 10.250.5.39 closed.
root@node-1:~# k get po
NAME                READY   STATUS    RESTARTS   AGE
webtool-wk8s-node-1 1/1     Running   0           11s
root@node-1:~#
```

#### Question #:45

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

See the solution below.

#### 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 -
```

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
```

#### Question #:46

Given a partially-functioningKubernetes cluster, identifysymptoms of failure on the cluster.

Determine the node, the failingservice, and take actions to bring upthe failed service and restore thehealth of the cluster. Ensure that anychanges are made permanently.

You canssh to the relevant Inodes (*bk8s-master-0*or*bk8s-node-0*) using:

```
[student@node-1] $ ssh<nodename>
```

You can assume elevatedprivileges on any node in thecluster with the followingcommand:

```
[student@nodename] $ | sudo -i
```

See the solutionbelow.

### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\23 C.JPG





The screenshot shows a web terminal interface with a dark blue header. On the left, there are two buttons: 'Readme' with a book icon and 'Web Terminal' with a terminal icon. On the right, the 'THE LINUX FOUNDATION' logo is displayed. The terminal window has a black background with white text. The user is logged in as 'root' on a machine named 'node-1'. They execute 'kubectl config use-context bk8s', which switches the context to 'bk8s'. Then, they execute 'ssh bk8s-master-0', connecting to another machine. The new prompt is 'root@bk8s-master-0:~#'. The terminal shows Ubuntu 16.04.6 LTS information and several system messages about updates and documentation. Finally, the user runs 'sudo -i' to become root, and the prompt changes to 'root@bk8s-master-0:~#'. They then start editing the file '/var/lib/kubelet/config.yaml' using the 'vim' editor.

```
root@node-1:~#
root@node-1:~# kubectl config use-context bk8s
Switched to context "bk8s".
root@node-1:~# ssh bk8s-master-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@bk8s-master-0:~$ sudo -i
root@bk8s-master-0:~# vim /var/lib/kubelet/config.yaml
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\23 D.JPG

A screenshot of a web terminal interface. At the top, there is a dark blue header bar. On the left, there are two tabs: 'Readme' (with a book icon) and 'Web Terminal' (with a terminal icon). On the right, the text 'THE LINUX FOUNDATION' is displayed in white. The main area of the terminal is black with white text. It shows a list of configuration parameters for a Kubernetes component, likely kubelet. The parameters include authorization mode (Webhook), cache TTLs, cluster DNS, domain, CPU manager reconcile period, eviction pressure transition period, file check frequency, healthz bind address and port, HTTP check frequency, image minimum GC age, kind (KubeletConfiguration), node status report and update frequencies, rotate certificates (true), runtime request timeout, static pod path, streaming connection idle timeout, sync frequency, and volume stats aggregation period. The text is wrapped, and a vertical scrollbar is visible on the right side of the terminal window.

```
authorization:
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
:wc
```

F:\Work\Data Entry Work\Data Entry\abc\CKA\23 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@bk8s-master-0:~$ sudo -i
root@bk8s-master-0:~# vim /var/lib/kubelet/config.yaml
root@bk8s-master-0:~# systemctl restart kubelet
root@bk8s-master-0:~# systemctl enable kubelet
root@bk8s-master-0:~# kubectl get nodes

NAME             STATUS    ROLES    AGE   VERSION
bk8s-master-0    Ready    master   77d   v1.18.2
bk8s-node-0      Ready    <none>   77d   v1.18.2
root@bk8s-master-0:~#
root@bk8s-master-0:~# exit
logout
student@bk8s-master-0:~$ exit
logout
Connection to 10.250.4.77 closed.
root@node-1:~#
```

#### Question #:47

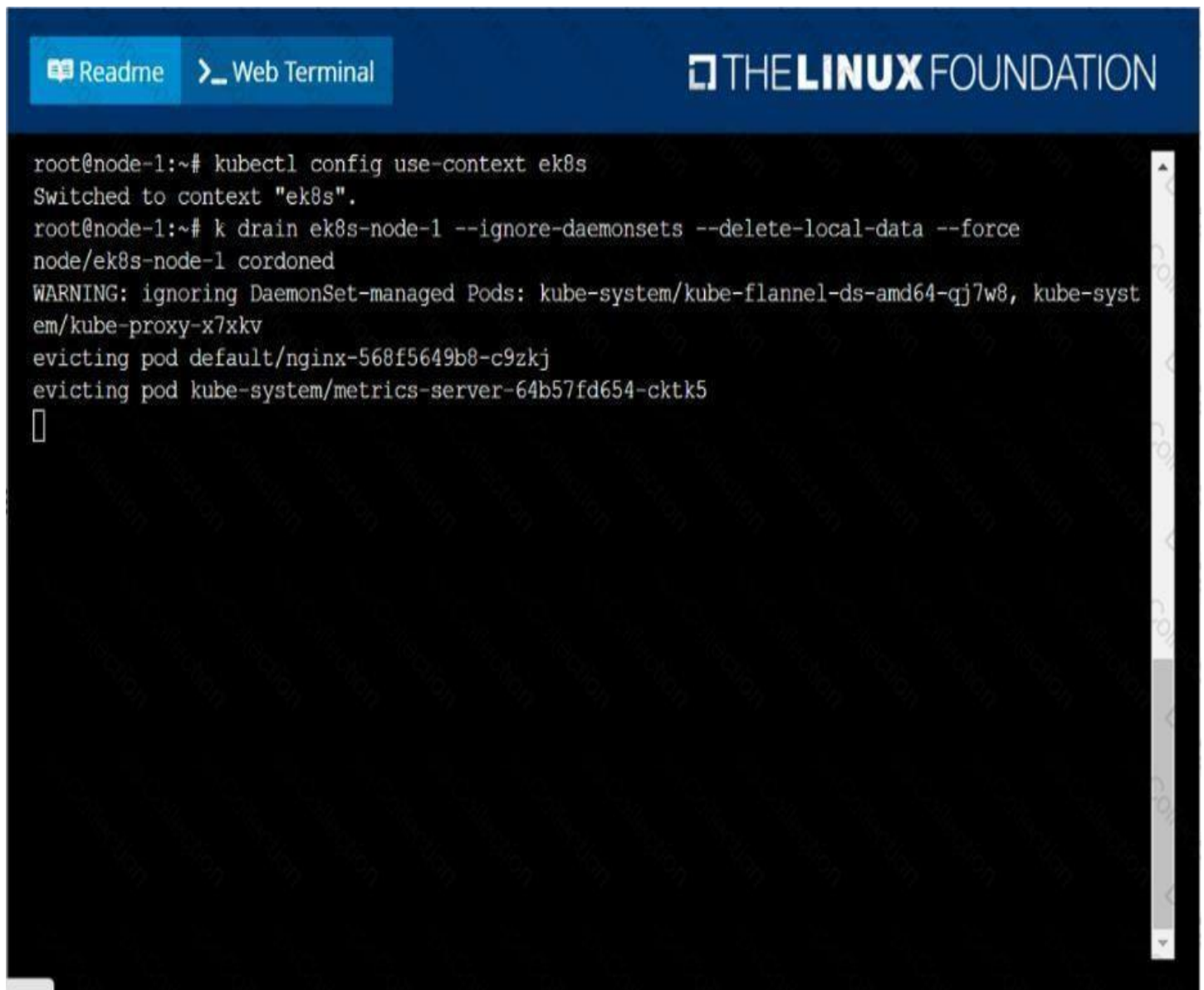
Set the node named *bk8s-node-1* as unavailable and reschedule all the pods running on it.

See the solution below.

#### Explanation

solution

F:\Work\Data Entry Work\Data Entry\abc\CKA\19 B.JPG



```
root@node-1:~# kubectl config use-context ek8s
Switched to context "ek8s".
root@node-1:~# k drain ek8s-node-1 --ignore-daemonsets --delete-local-data --force
node/ek8s-node-1 cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-flannel-ds-amd64-qj7w8, kube-system/kube-proxy-x7xkv
evicting pod default/nginx-568f5649b8-c9zkj
evicting pod kube-system/metrics-server-64b57fd654-cktk5
[]
```

#### Question #:48

Check the Image version of nginx-dev pod using jsonpath

See the solution below.

#### Explanation

```
kubectl get po nginx-dev -o
```

```
jsonpath='{.spec.containers[].image} {"\n"}'
```

