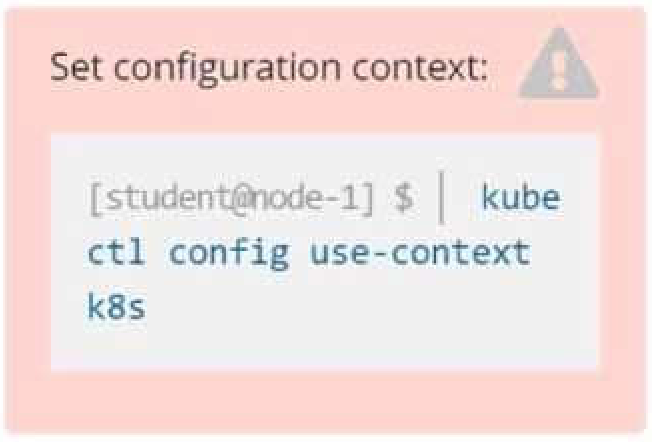
Mirantis CKA : Practice Test

Exam Code: CKA

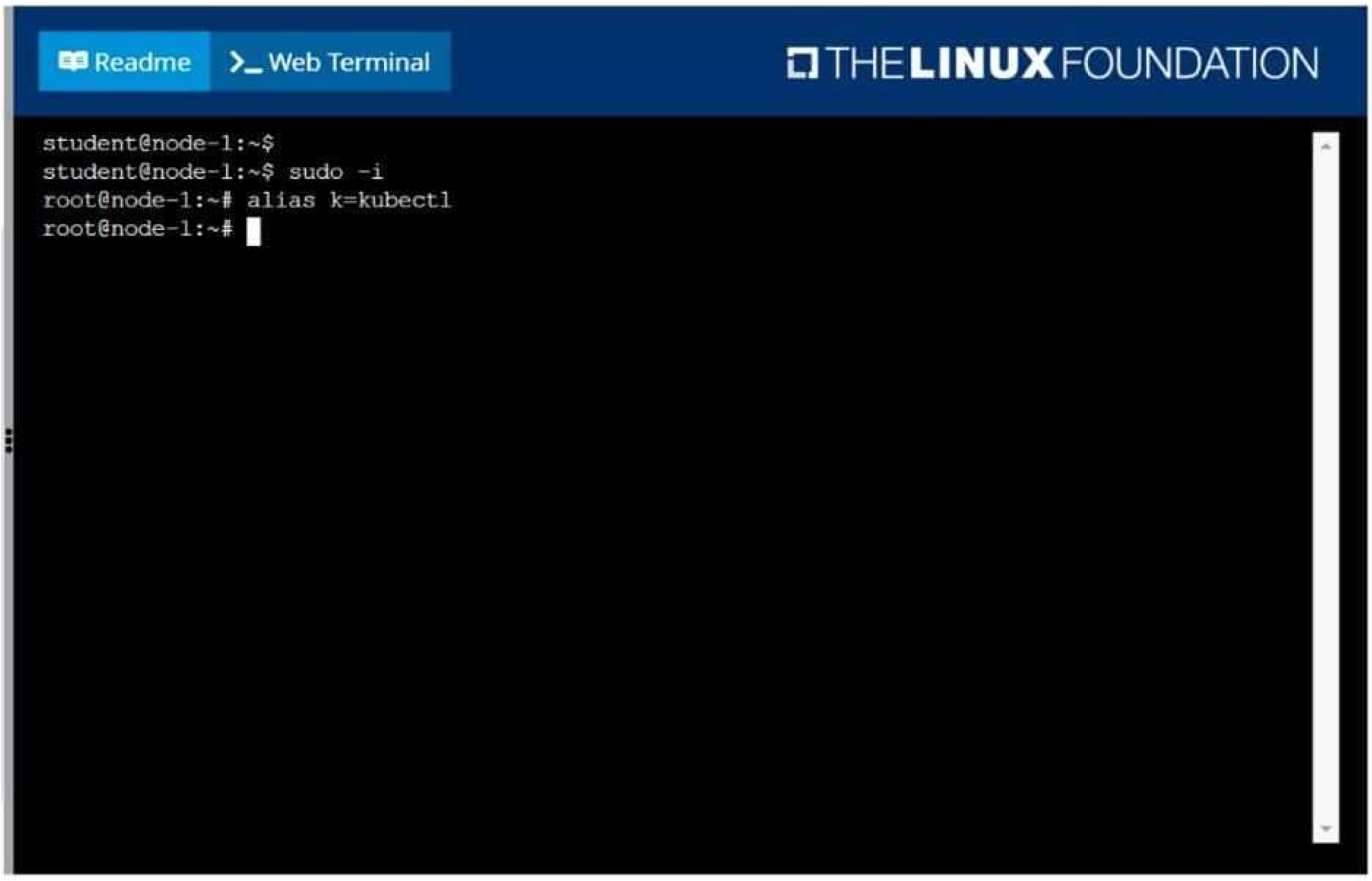
Title : Certified Kubernetes Administrator

QUESTION 1
SIMULATION
Monitor the logs of pod foo and:
· Extract log lines corresponding to error unable-to-access-website
· Write them to /opt/KULM00201/foo

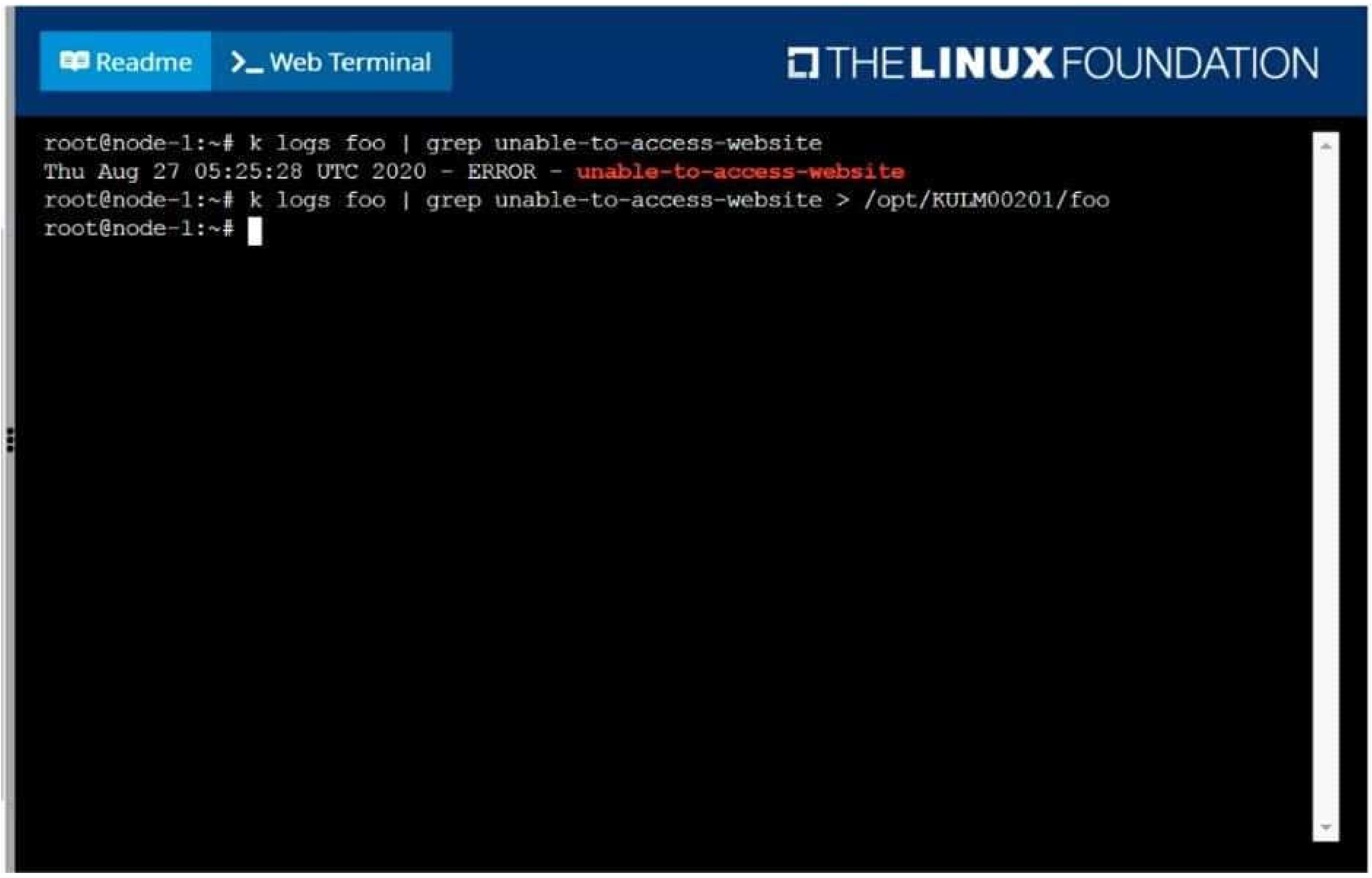

Set configuration context:

```
[student@node-1] $ | kubectl config use-context k8s
```

A.



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



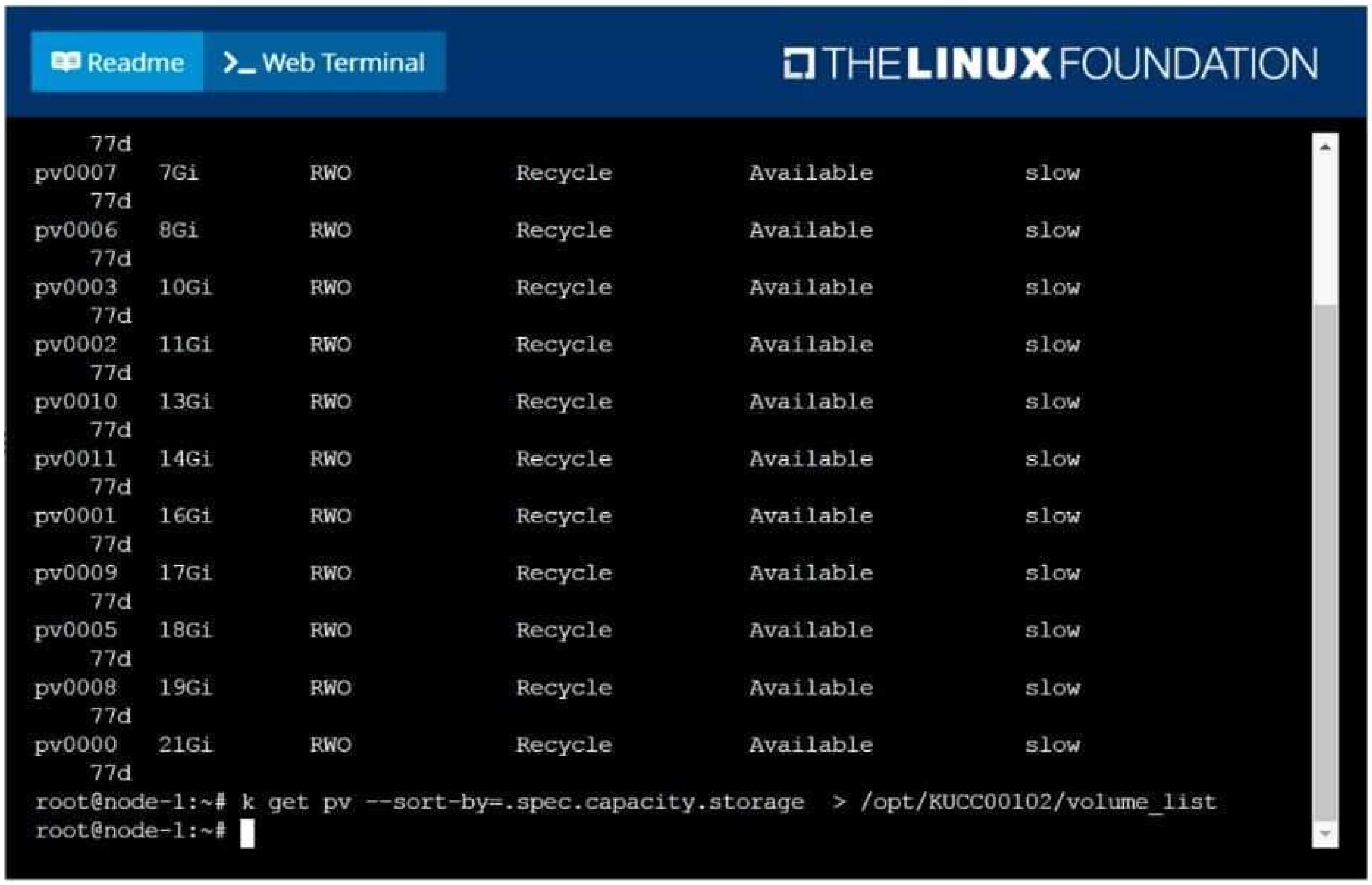
```
root@node-1:~# k logs foo | grep unable-to-access-website
Thu Aug 27 05:25:28 UTC 2020 - ERROR - unable-to-access-website
root@node-1:~# k logs foo | grep unable-to-access-website > /opt/KULM00201/foo
root@node-1:~#
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 2
SIMULATION
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.



```
    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:~#
```

Correct Answer: A

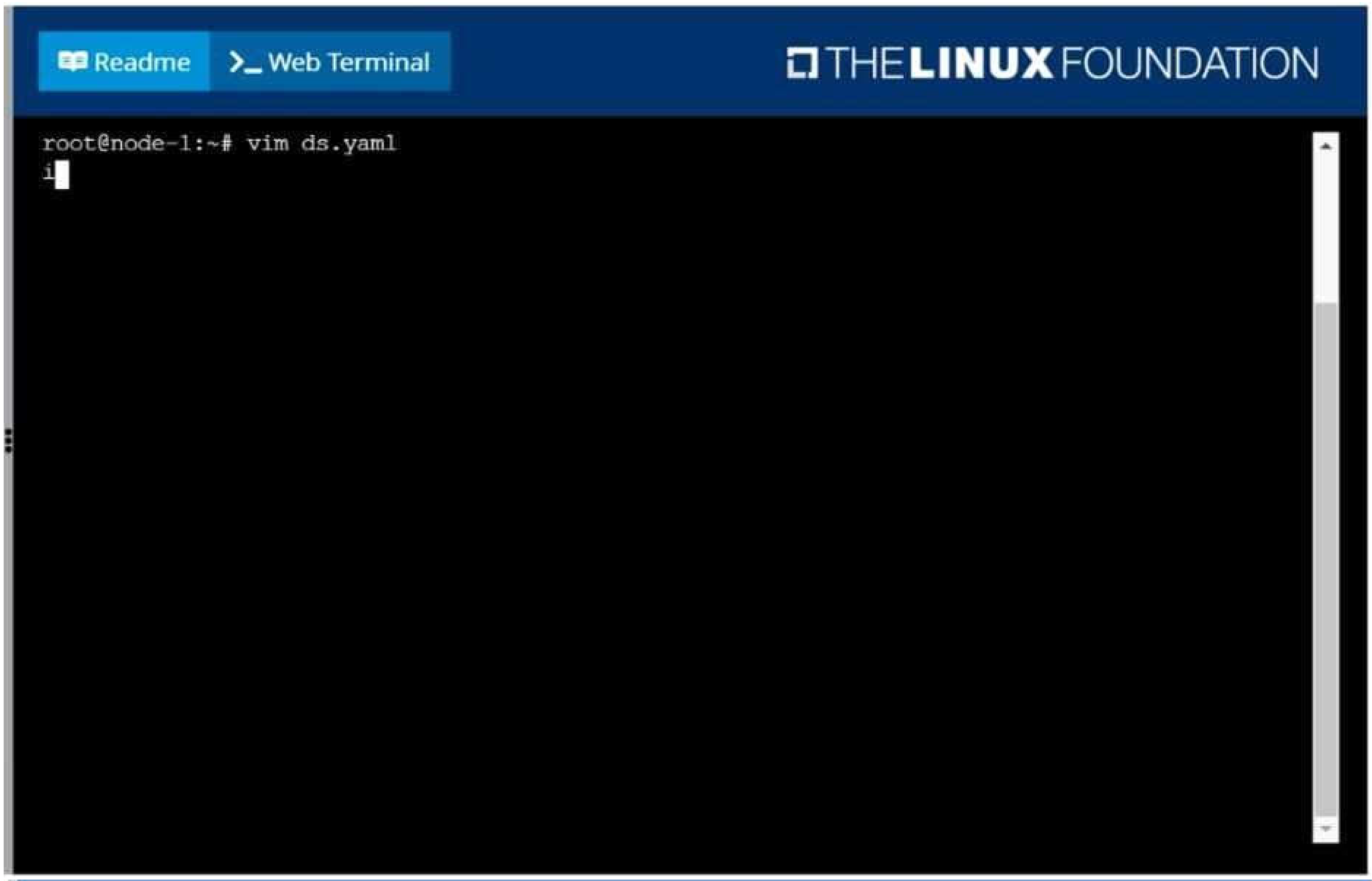
Explanation

Explanation/Reference:

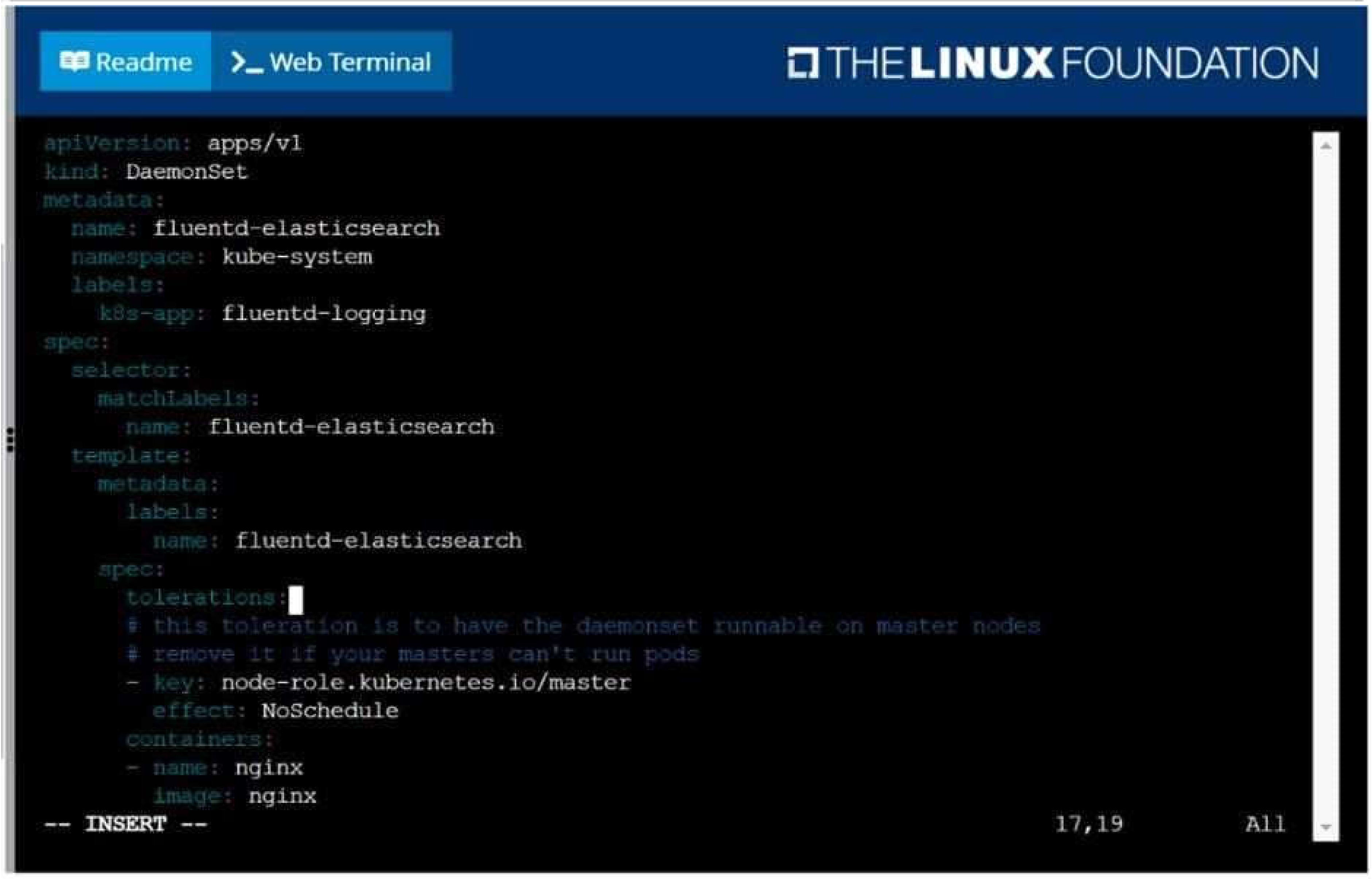
QUESTION 3
SIMULATION
Ensure a single instance of pod nginx is running on each node of the Kubernetes cluster where nginx also represents the Image name which has to be used. Do not override any taints currently in place.

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

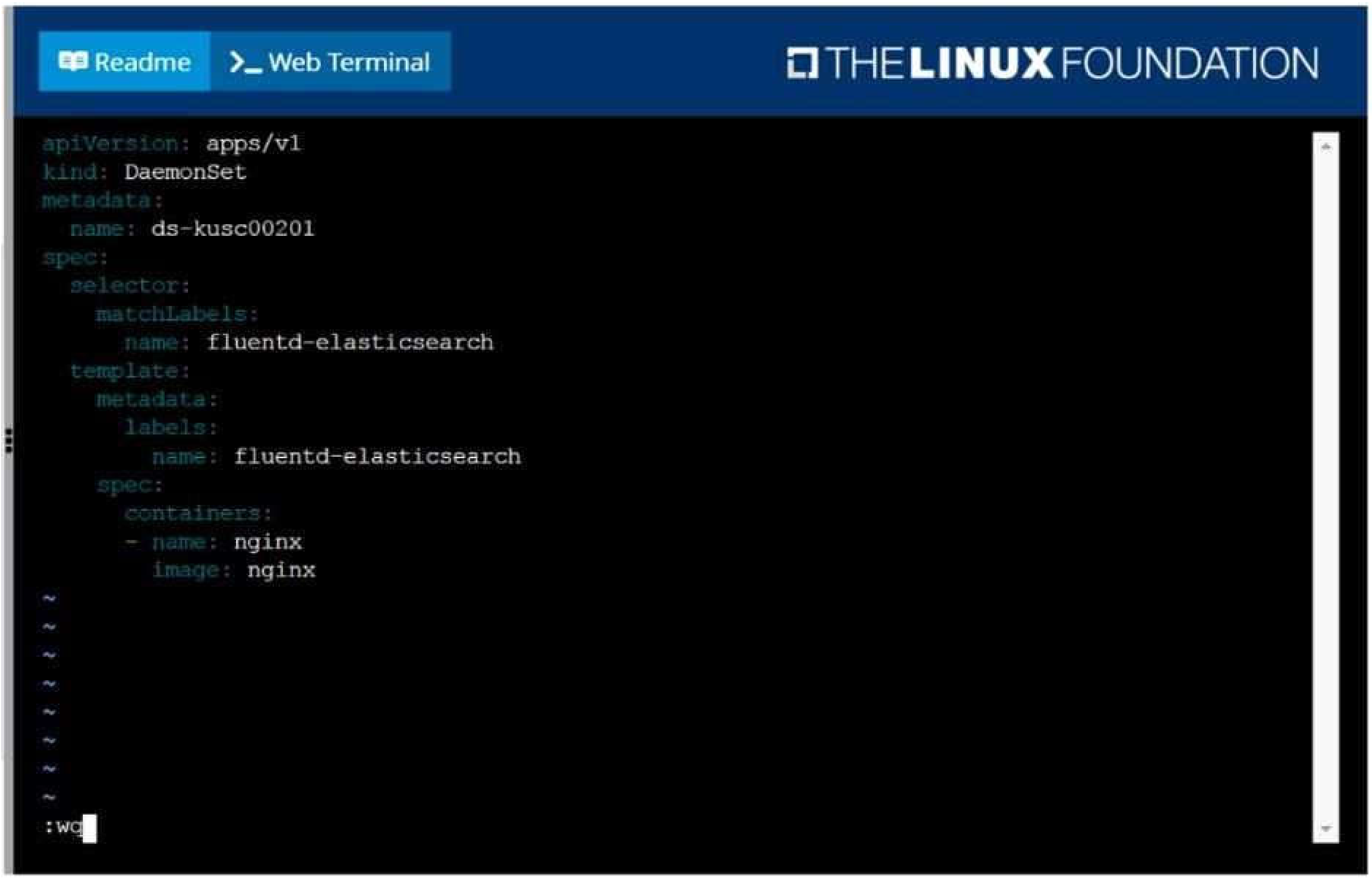
A.



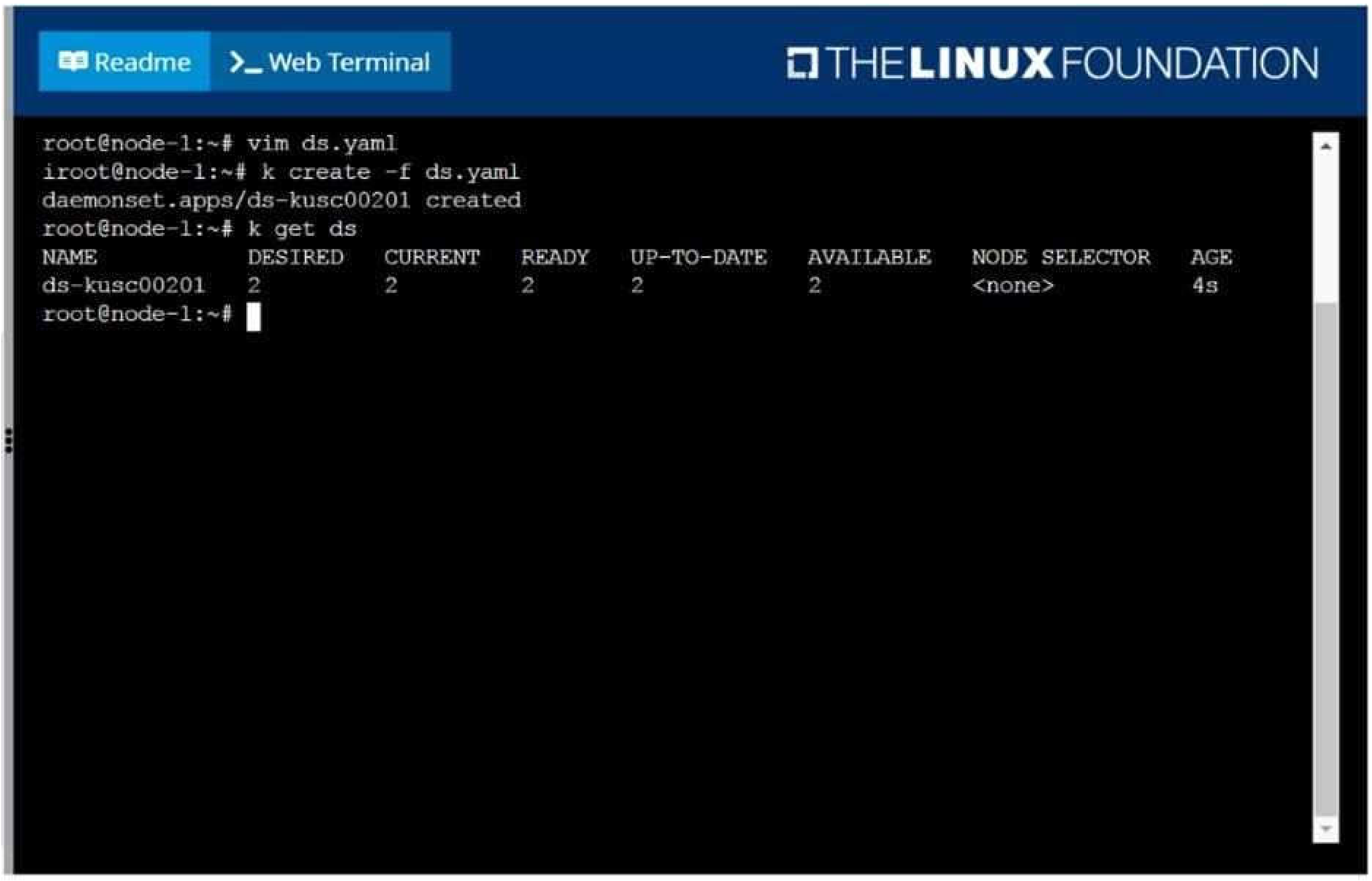
```
root@node-1:~# vim ds.yaml
i
```



```
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 --                                                17,19         All
```



```
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
~
~
~
~
~
~
~
~
:wq
```



```
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:~#
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 4
List all the pods sorted by created timestamp

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

Correct Answer: A

Explanation

Explanation/Reference:

QUESTION 5
List all the pods showing name and namespace with a json path expression

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

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 6
SIMULATION

Perform the following tasks:

* Add an init container to hungry-bear (which has been defined in spec file /opt/KUCC00108/pod-spec-KUCC00108.yaml)

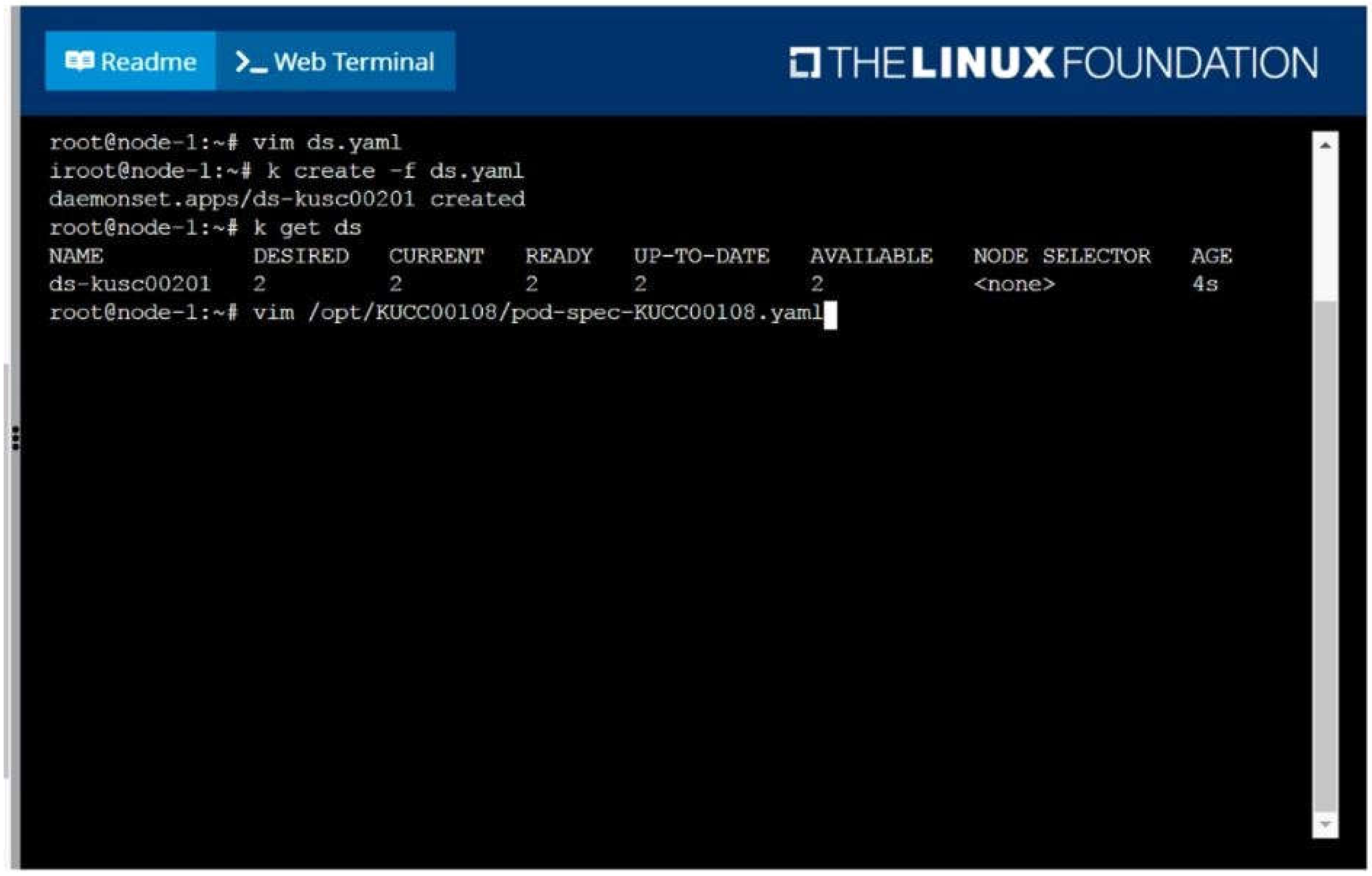
* The init container should create an empty file named

/workdir/calm.txt

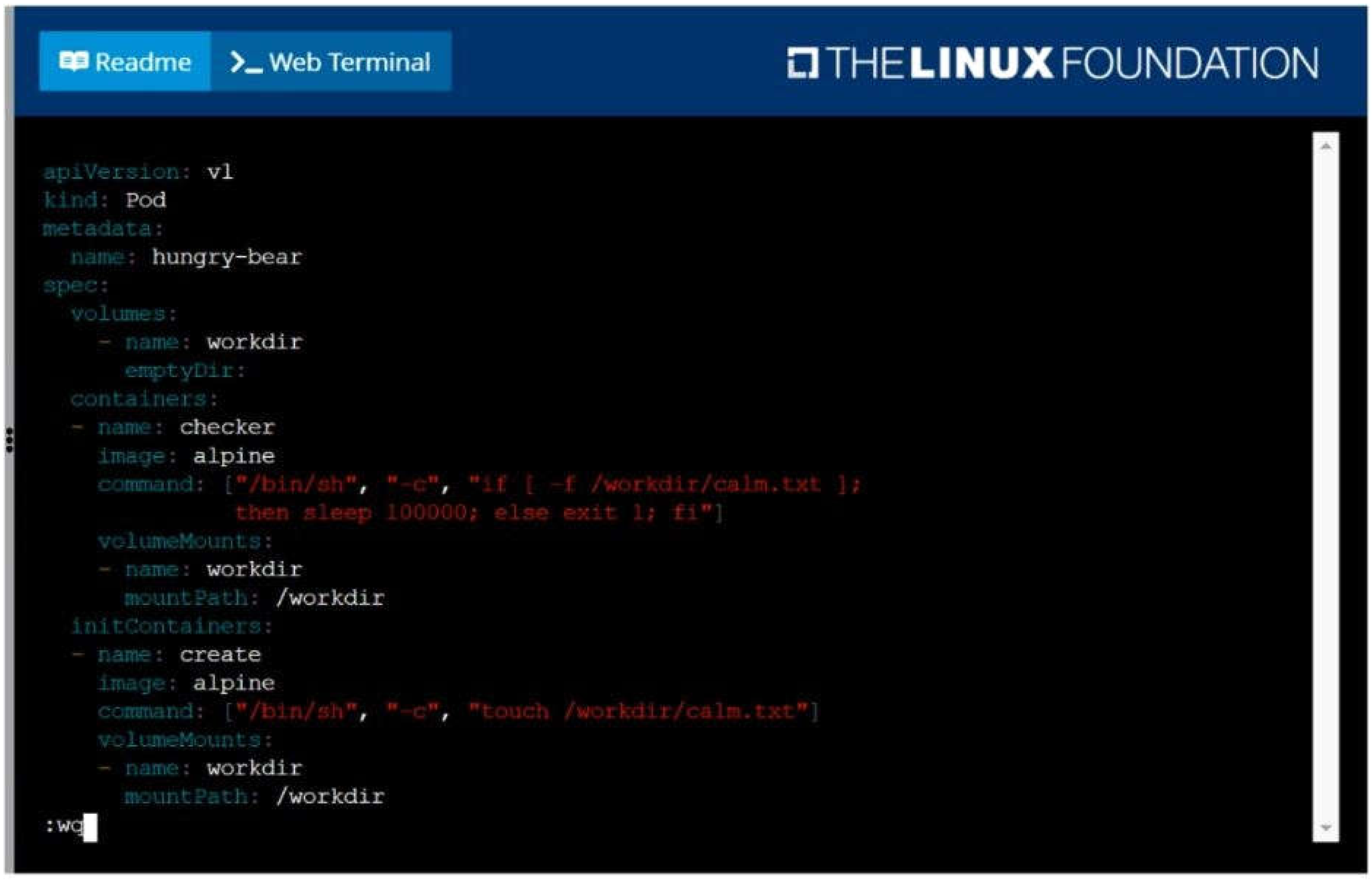
* If /workdir/calm.txt is not detected, the pod should exit

* Once the spec file has been updated with the init container definition, the pod should be created

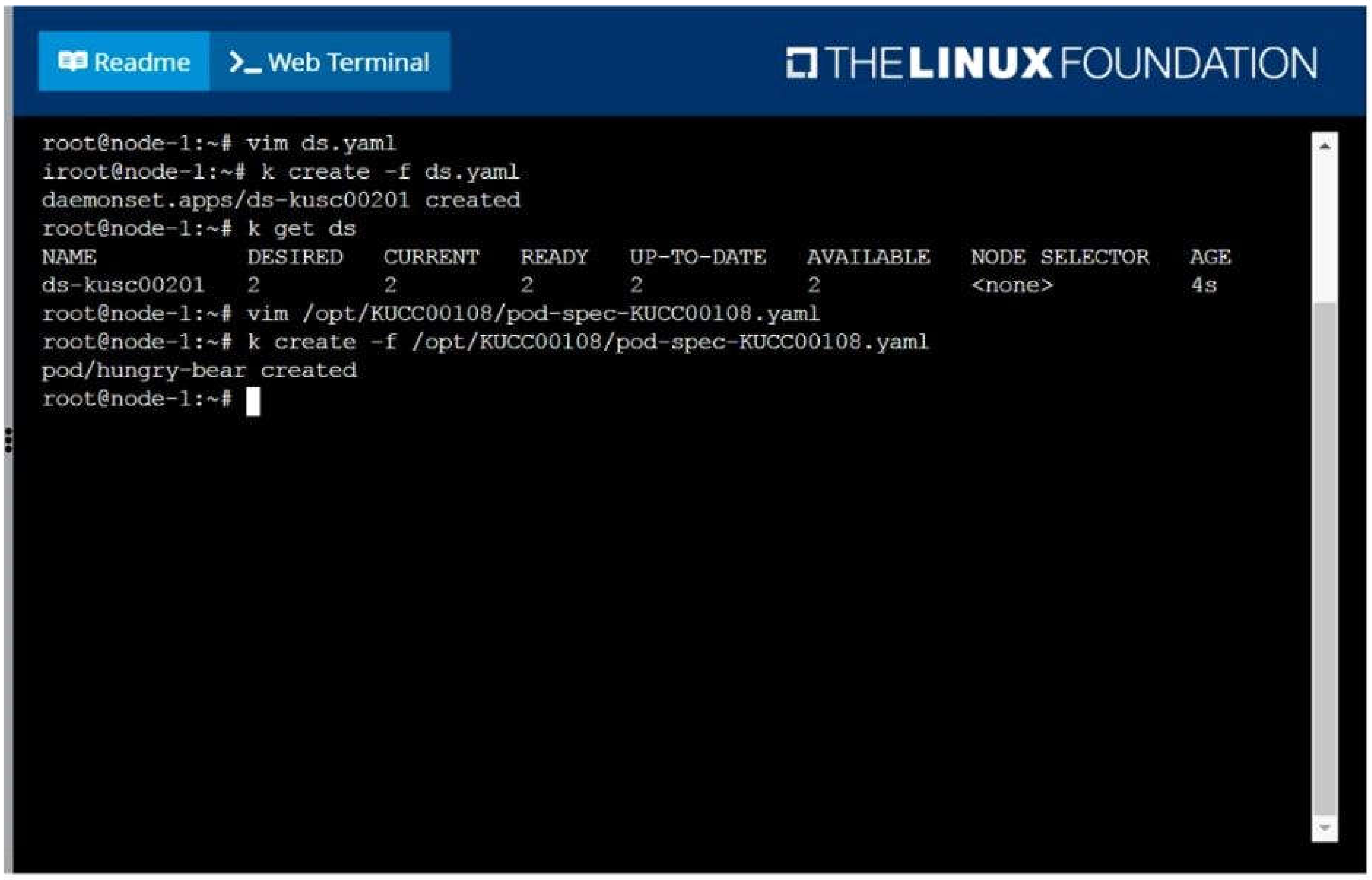
A.



```
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
```



```
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
```



```
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:~#
```

Correct Answer: A
Explanation

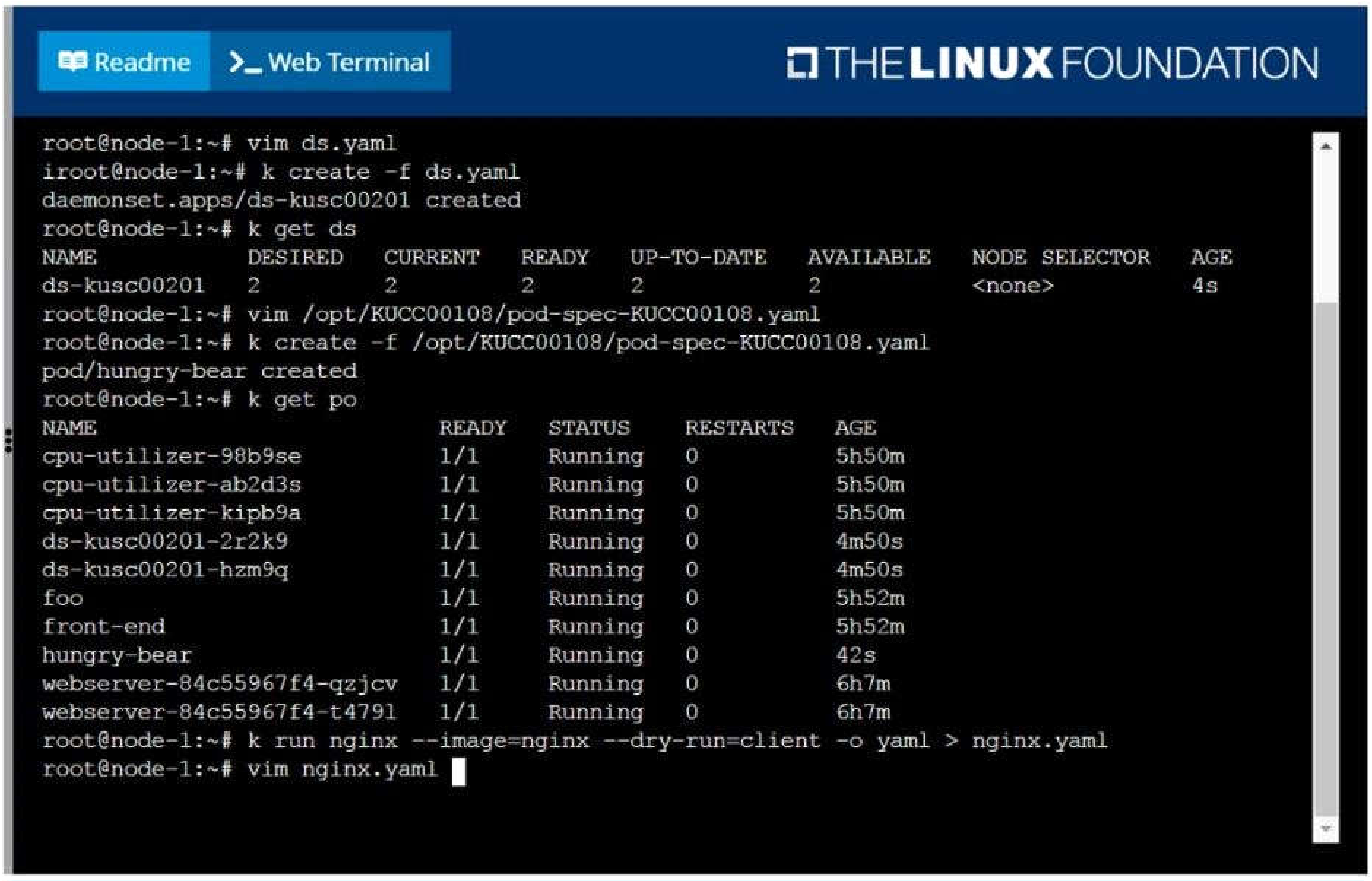
Explanation/Reference:

QUESTION 7
SIMULATION

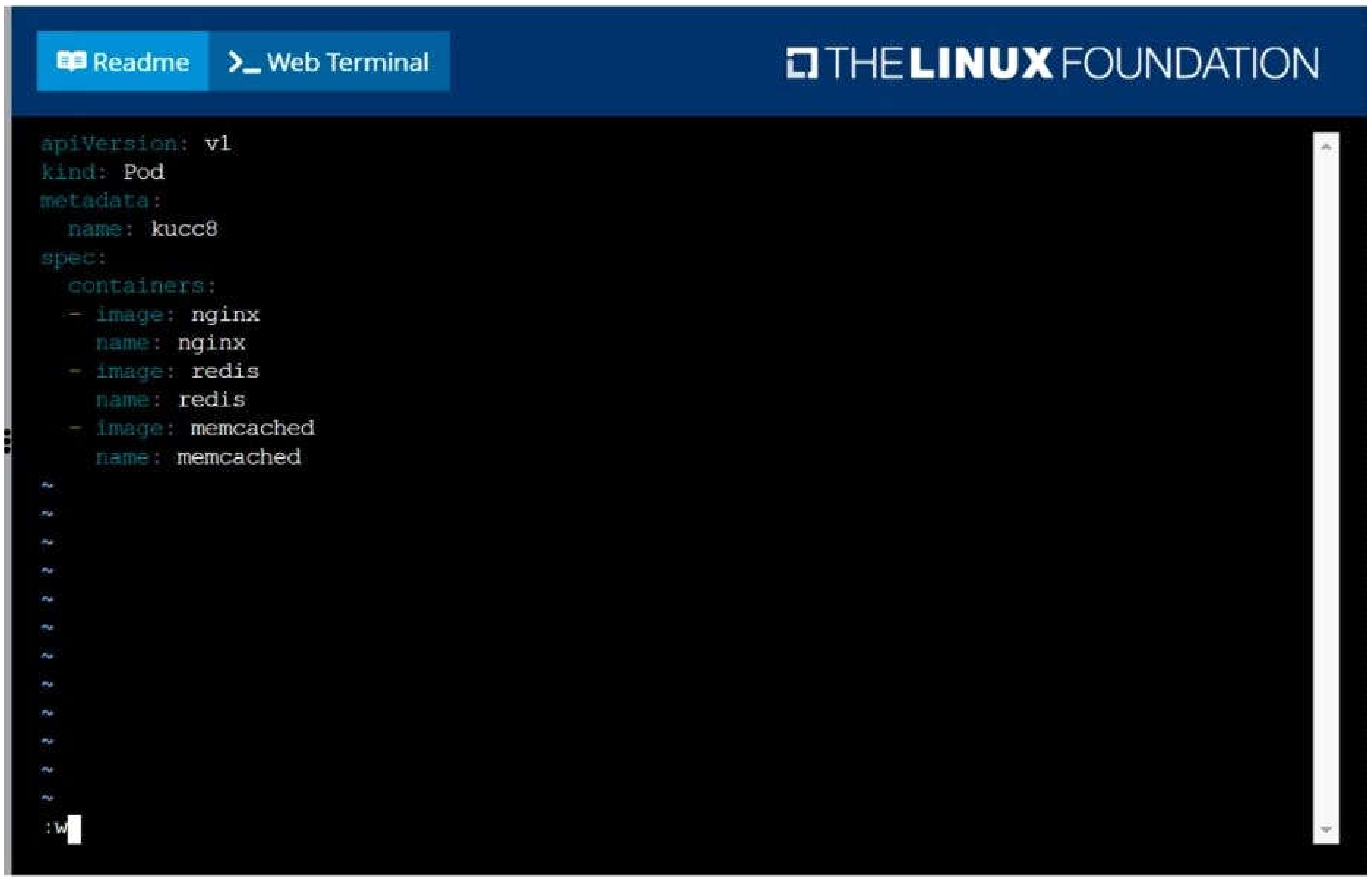
Create a pod named kucc8 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified):

nginx + redis + memcached.

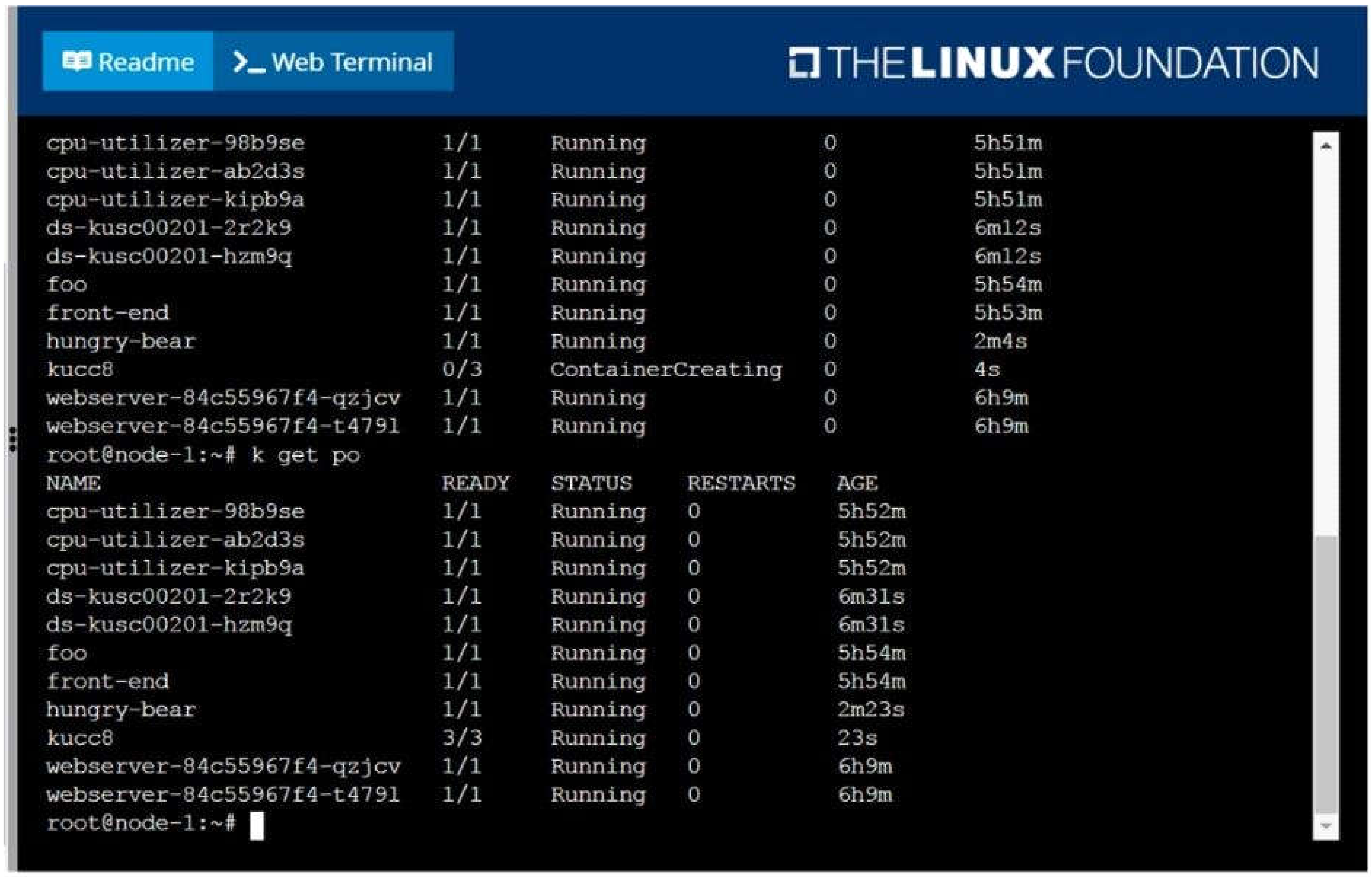
A.



```
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-t4791   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
```



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



```
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:~#
```

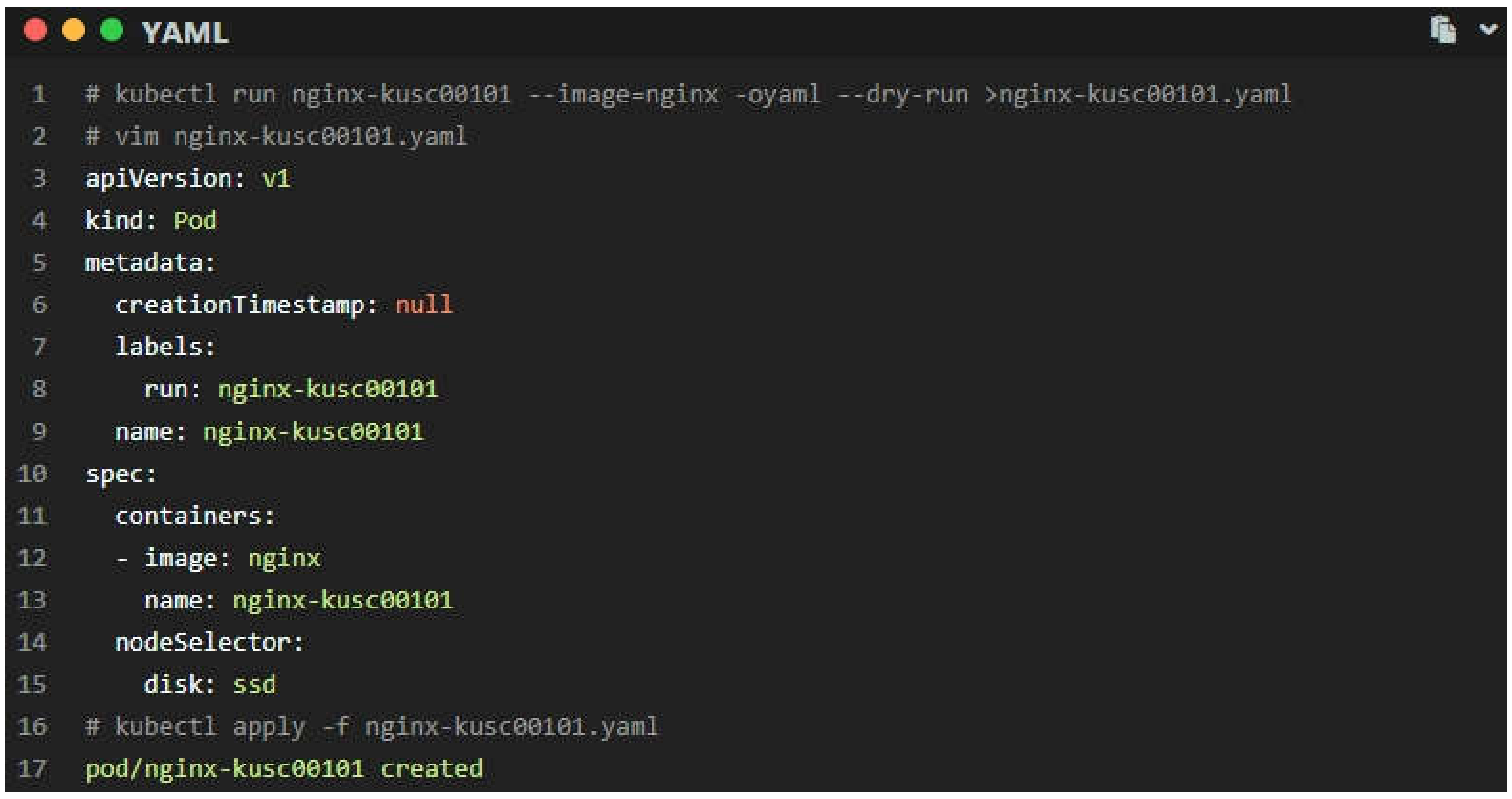
Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 8
Set configuration context $ kubectl config use-context k8s

Schedule a Pod as follows:
Name: nginx-kusc00101
Image: nginx
Node selector: disk=ssd

A.



```yaml
# kubectl run nginx-kusc00101 --image=nginx -oyaml --dry-run >nginx-kusc00101.yaml
# vim nginx-kusc00101.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx-kusc00101
  name: nginx-kusc00101
spec:
  containers:
  - image: nginx
    name: nginx-kusc00101
  nodeSelector:
    disk: ssd
# kubectl apply -f nginx-kusc00101.yaml
pod/nginx-kusc00101 created
```

Correct Answer: A
Explanation

Explanation/Reference:
https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

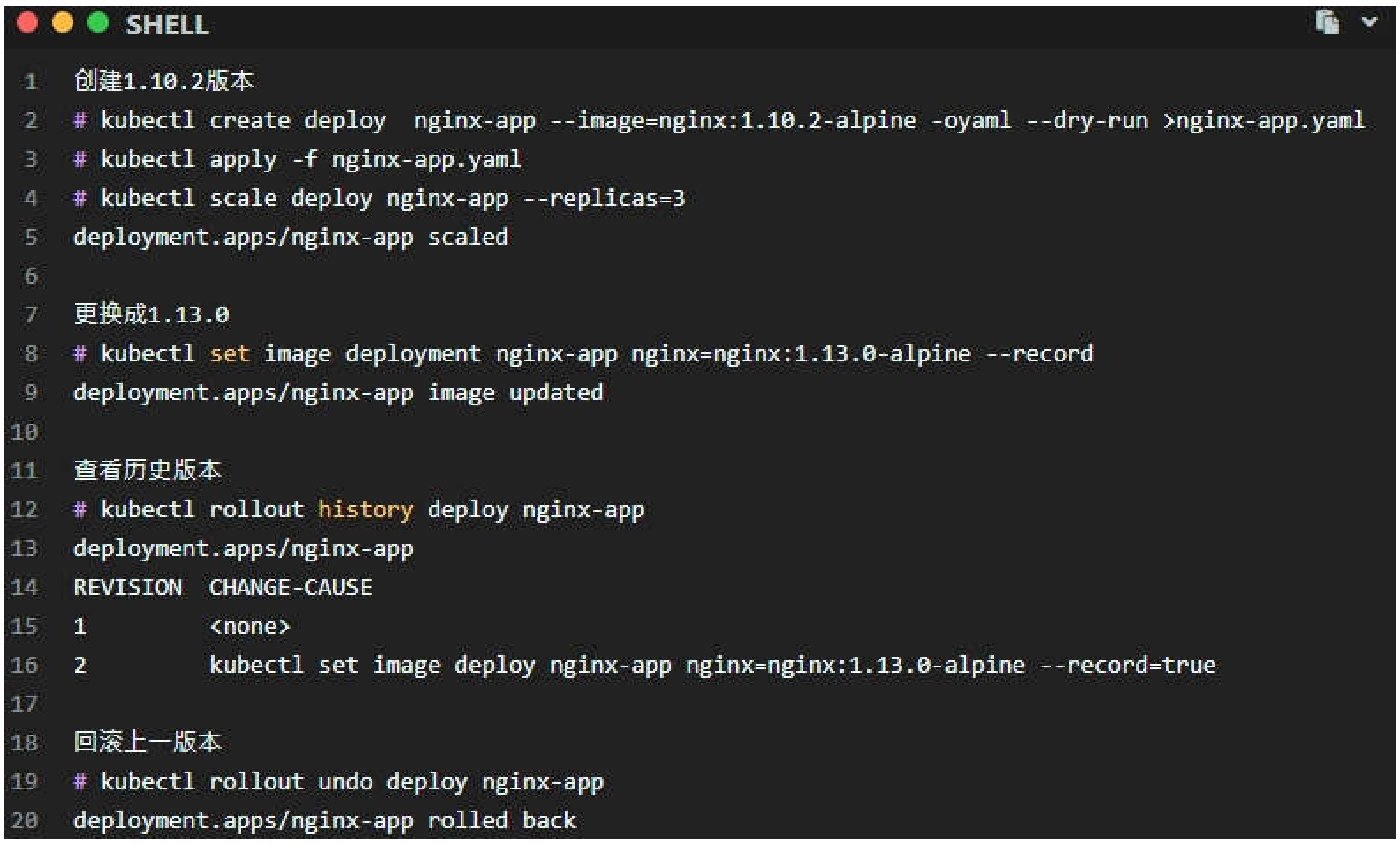
QUESTION 9
Set configuration context $ kubectl config use-context k8s

Create a deployment as follows
Name: nginx-app

Using container nginx with version 1.10.2-alpine
The deployment should contain 3 replicas
Next, deploy the app with new version 1.13.0-alpine by performing a rolling update and record that update.

Finally, rollback that update to the previous version 1.10.2-alpine

A.



SHELL

```
创建1.10.2版本
# kubectl create deploy  nginx-app --image=nginx:1.10.2-alpine -oyaml --dry-run >nginx-app.yaml
# kubectl apply -f nginx-app.yaml
# kubectl scale deploy nginx-app --replicas=3
deployment.apps/nginx-app scaled

更换成1.13.0
# kubectl set image deployment nginx-app nginx=nginx:1.13.0-alpine --record
deployment.apps/nginx-app image updated

查看历史版本
# kubectl rollout history deploy nginx-app
deployment.apps/nginx-app
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deploy nginx-app nginx=nginx:1.13.0-alpine --record=true

回滚上一版本
# kubectl rollout undo deploy nginx-app
deployment.apps/nginx-app rolled back
```

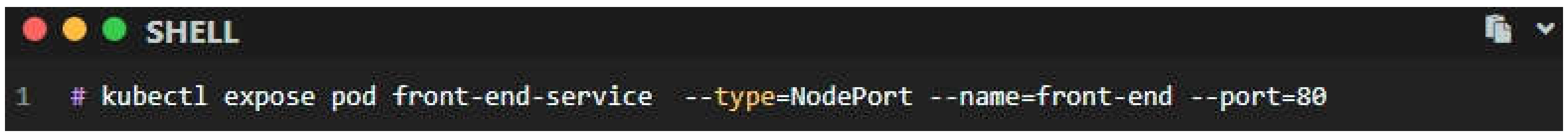
Correct Answer: A
Explanation

Explanation/Reference:
https://kubernetes.io/docs/reference/kubectl/cheatsheet/#updating-resources

QUESTION 10
Set configuration context $ kubectl config use-context k8s

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

A.



SHELL

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

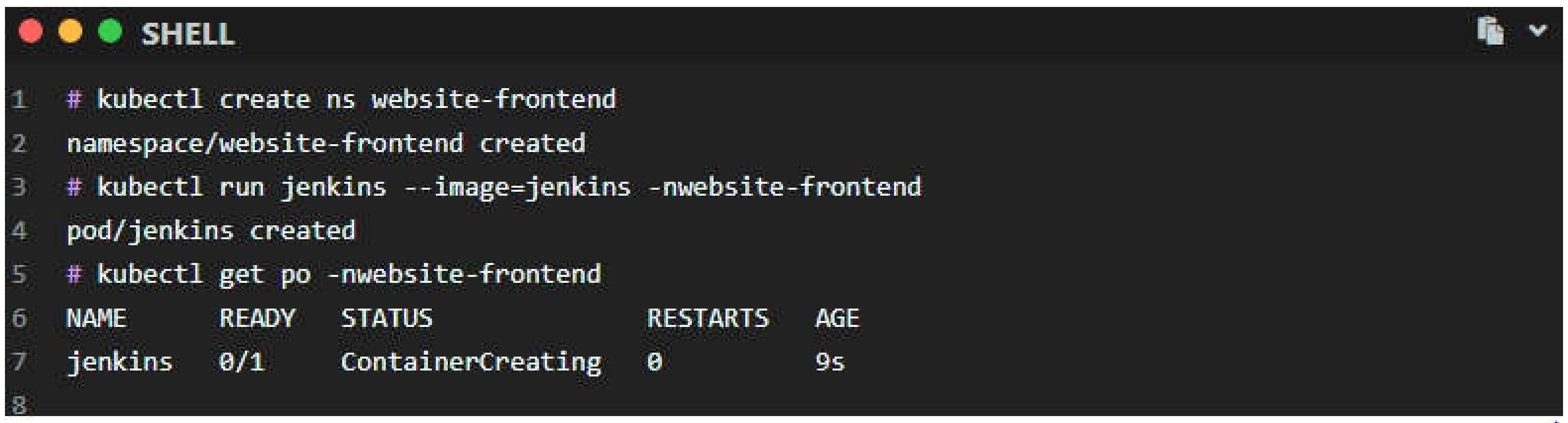
Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 11
Set configuration context $ kubectl config use-context k8s

Create a Pod as follows:
Name: jenkins
Using image: jenkins
In a new Kubenetes namespace named website-frontend

A.



SHELL

```
# kubectl create ns website-frontend
namespace/website-frontend created
# kubectl run jenkins --image=jenkins -nwebsite-frontend
pod/jenkins created
# kubectl get po -nwebsite-frontend
NAME      READY   STATUS              RESTARTS   AGE
jenkins   0/1     ContainerCreating   0          9s

```

Correct Answer: A
Explanation

Explanation/Reference:

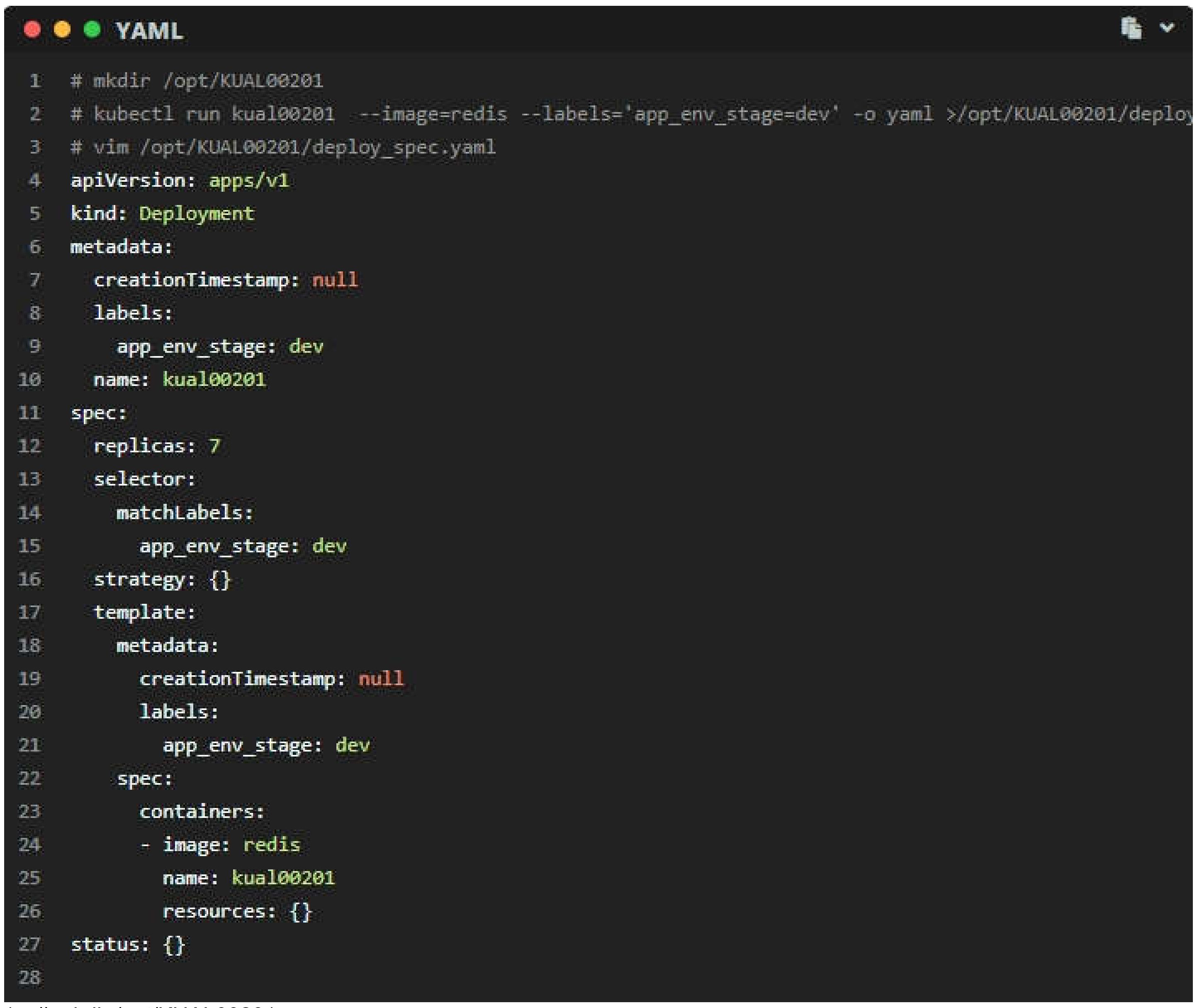
QUESTION 12
Set configuration context $ kubectl config use-context k8s

Create a deployment spec file that will:

Launch 7 replicas of the redis image with the label: app_env_stage=dev
Deployment name: kual00201
Save a copy of this spec file to /opt/KUAL00201/deploy_spec.yaml (or .json)

When you are done, clean up (delete) any new k8s API objects that you produced during this task

A.



```yaml
# mkdir /opt/KUAL00201
# kubectl run kual00201  --image=redis --labels='app_env_stage=dev' -o yaml >/opt/KUAL00201/deploy
# vim /opt/KUAL00201/deploy_spec.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app_env_stage: dev
  name: kual00201
spec:
  replicas: 7
  selector:
    matchLabels:
      app_env_stage: dev
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app_env_stage: dev
    spec:
      containers:
      - image: redis
        name: kual00201
        resources: {}
status: {}
```

```yaml
# mkdir /opt/KUAL00201
# kubectl run kual00201  --image=redis --labels='app_env_stage=dev' -o yaml >/opt/KUAL00201/deploy_spec.yaml
# vim /opt/KUAL00201/deploy_spec.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app_env_stage: dev
  name: kual00201
spec:
  replicas: 7
  selector:
    matchLabels:
      app_env_stage: dev
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app_env_stage: dev
    spec:
      containers:
      - image: redis
        name: kual00201
        resources: {}
status: {}
```

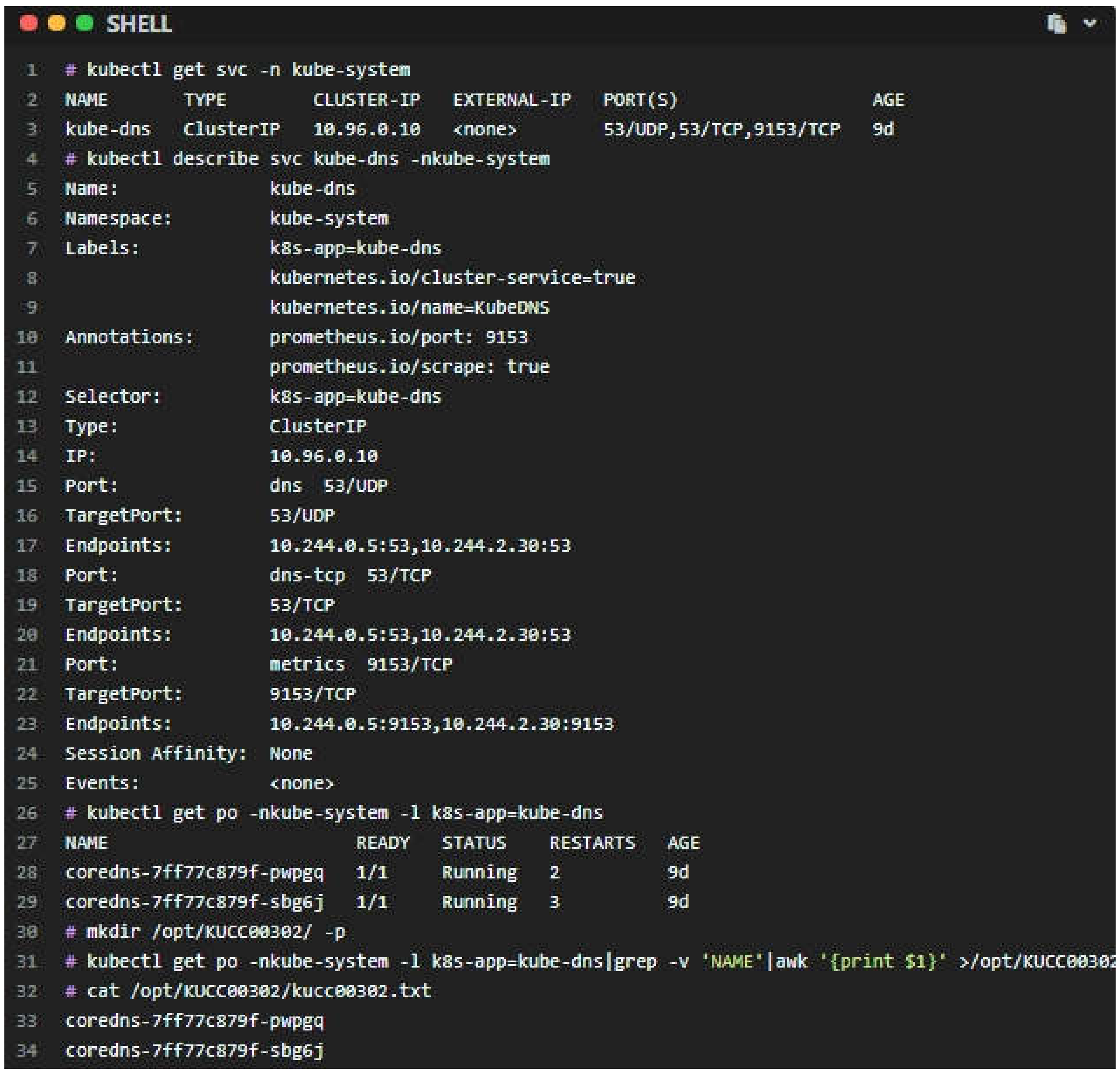
Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 13
Set configuration context $ kubectl config use-context k8s

Create a file /opt/KUCC00302/kucc00302.txt that lists all pods that implement Service foo in Namespace production.

The format of the file should be one pod name per line.

A. 
SHELL

```
# kubectl get svc -n kube-system
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.96.0.10   <none>        53/UDP,53/TCP,9153/TCP   9d
# kubectl describe svc kube-dns -nkube-system
Name:              kube-dns
Namespace:         kube-system
Labels:            k8s-app=kube-dns
                   kubernetes.io/cluster-service=true
                   kubernetes.io/name=KubeDNS
Annotations:       prometheus.io/port: 9153
                   prometheus.io/scrape: true
Selector:          k8s-app=kube-dns
Type:              ClusterIP
IP:                10.96.0.10
Port:              dns  53/UDP
TargetPort:        53/UDP
Endpoints:         10.244.0.5:53,10.244.2.30:53
Port:              dns-tcp  53/TCP
TargetPort:        53/TCP
Endpoints:         10.244.0.5:53,10.244.2.30:53
Port:              metrics  9153/TCP
TargetPort:        9153/TCP
Endpoints:         10.244.0.5:9153,10.244.2.30:9153
Session Affinity:  None
Events:            <none>
# kubectl get po -nkube-system -l k8s-app=kube-dns
NAME                       READY   STATUS    RESTARTS   AGE
coredns-7ff77c879f-pwpgq   1/1     Running   2          9d
coredns-7ff77c879f-sbg6j   1/1     Running   3          9d
# mkdir /opt/KUCC00302/ -p
# kubectl get po -nkube-system -l k8s-app=kube-dns|grep -v 'NAME'|awk '{print $1}' >/opt/KUCC0030
# cat /opt/KUCC00302/kucc00302.txt
coredns-7ff77c879f-pwpgq
coredns-7ff77c879f-sbg6j
```

Correct Answer:
Explanation

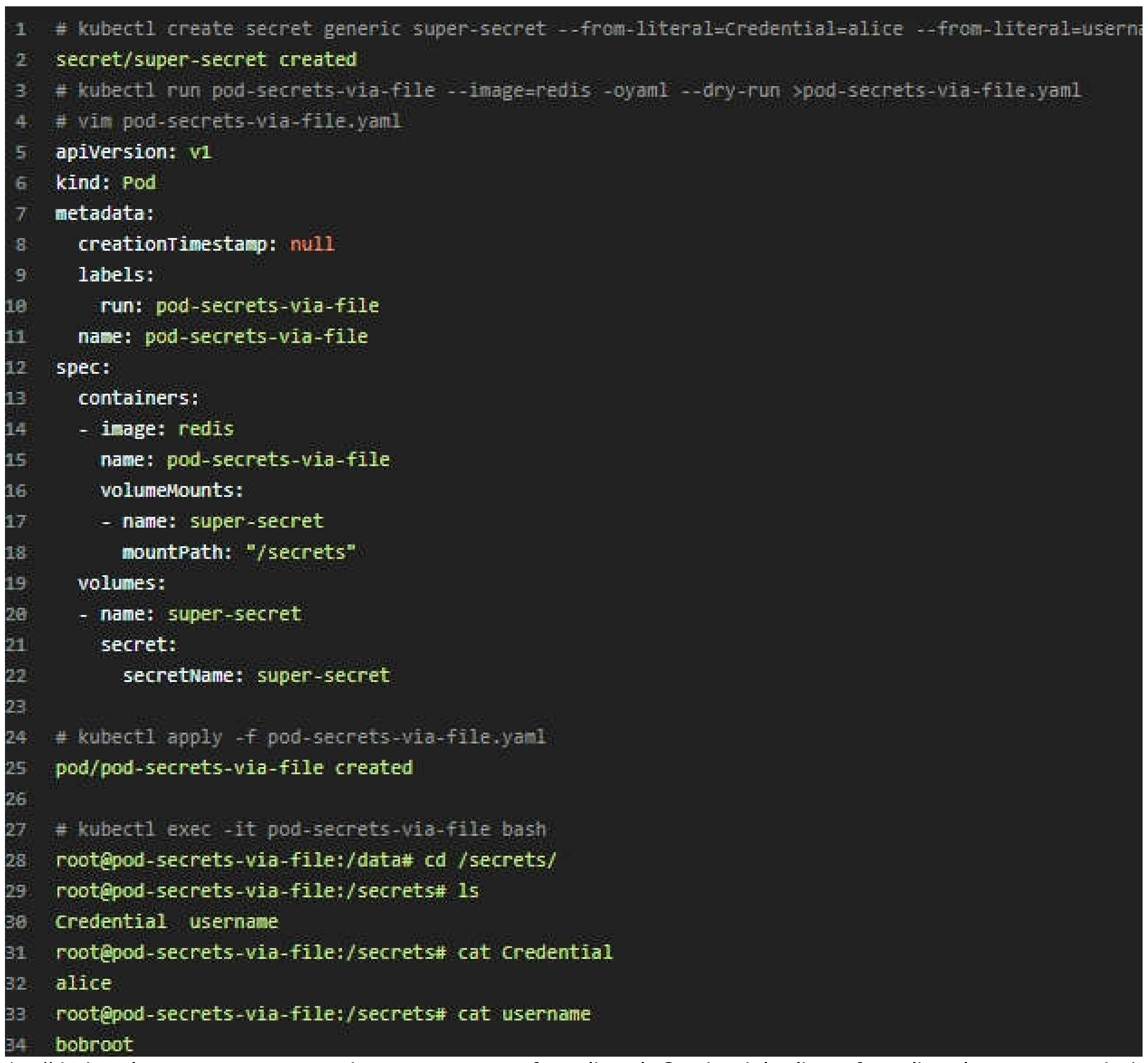
Explanation/Reference:

QUESTION 14
Set configuration context $ kubectl config use-context k8s

Create a Kubernetes Secret as follows:
Name: super-secret
Credential: alice or username:bob
Create a Pod named pod-secrets-via-file using the redis image which mounts a secret named super-secret at /secrets

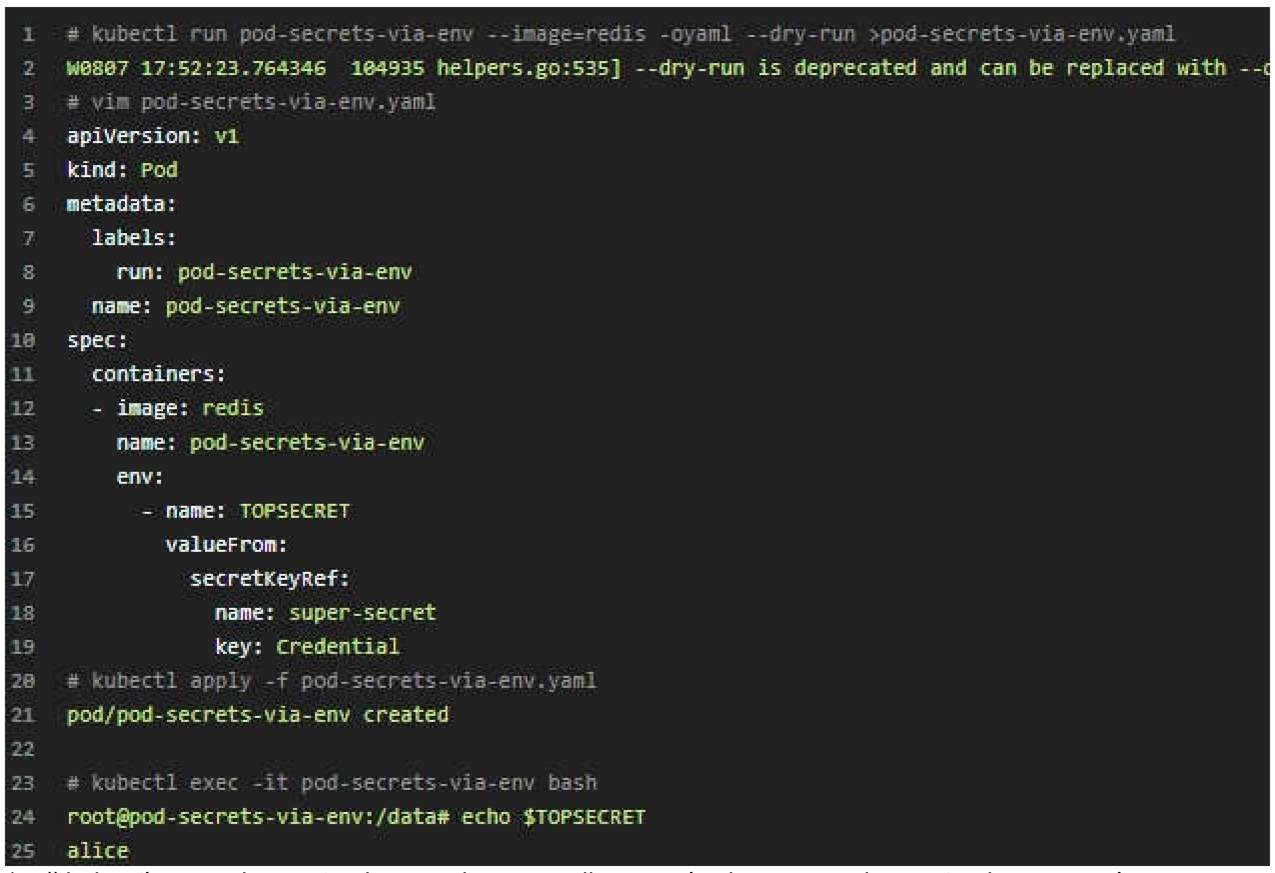
Create a second Pod named pod-secrets-via-env using the redis image, which exports credential as TOPSECRET

A.



```
# kubectl create secret generic super-secret --from-literal=Credential=alice --from-literal=userna
secret/super-secret created
# kubectl run pod-secrets-via-file --image=redis -oyaml --dry-run >pod-secrets-via-file.yaml
# vim pod-secrets-via-file.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: pod-secrets-via-file
  name: pod-secrets-via-file
spec:
  containers:
  - image: redis
    name: pod-secrets-via-file
    volumeMounts:
    - name: super-secret
      mountPath: "/secrets"
  volumes:
  - name: super-secret
    secret:
      secretName: super-secret

# kubectl apply -f pod-secrets-via-file.yaml
pod/pod-secrets-via-file created

# kubectl exec -it pod-secrets-via-file bash
root@pod-secrets-via-file:/data# cd /secrets/
root@pod-secrets-via-file:/secrets# ls
Credential  username
root@pod-secrets-via-file:/secrets# cat Credential
alice
root@pod-secrets-via-file:/secrets# cat username
bobroot
```

```
# kubectl create secret generic super-secret --from-literal=Credential=alice --from-literal=username=bob
secret/super-secret created
# kubectl run pod-secrets-via-file --image=redis -oyaml --dry-run >pod-secrets-via-file.yaml
# vim pod-secrets-via-file.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: pod-secrets-via-file
  name: pod-secrets-via-file
spec:
  containers:
  - image: redis
    name: pod-secrets-via-file
    volumeMounts:
    - name: super-secret
      mountPath: "/secrets"
  volumes:
  - name: super-secret
    secret:
      secretName: super-secret

# kubectl apply -f pod-secrets-via-file.yaml
pod/pod-secrets-via-file created

# kubectl exec -it pod-secrets-via-file bash
root@pod-secrets-via-file:/data# cd /secrets/
root@pod-secrets-via-file:/secrets# ls
Credential  username
root@pod-secrets-via-file:/secrets# cat Credential
alice
root@pod-secrets-via-file:/secrets# cat username
bobroot
```



```
# kubectl run pod-secrets-via-env --image=redis -oyaml --dry-run >pod-secrets-via-env.yaml
W0807 17:52:23.764346  104935 helpers.go:535] --dry-run is deprecated and can be replaced with --d
# vim pod-secrets-via-env.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod-secrets-via-env
  name: pod-secrets-via-env
spec:
  containers:
  - image: redis
    name: pod-secrets-via-env
    env:
      - name: TOPSECRET
        valueFrom:
          secretKeyRef:
            name: super-secret
            key: Credential
# kubectl apply -f pod-secrets-via-env.yaml
pod/pod-secrets-via-env created

# kubectl exec -it pod-secrets-via-env bash
root@pod-secrets-via-env:/data# echo $TOPSECRET
alice
```

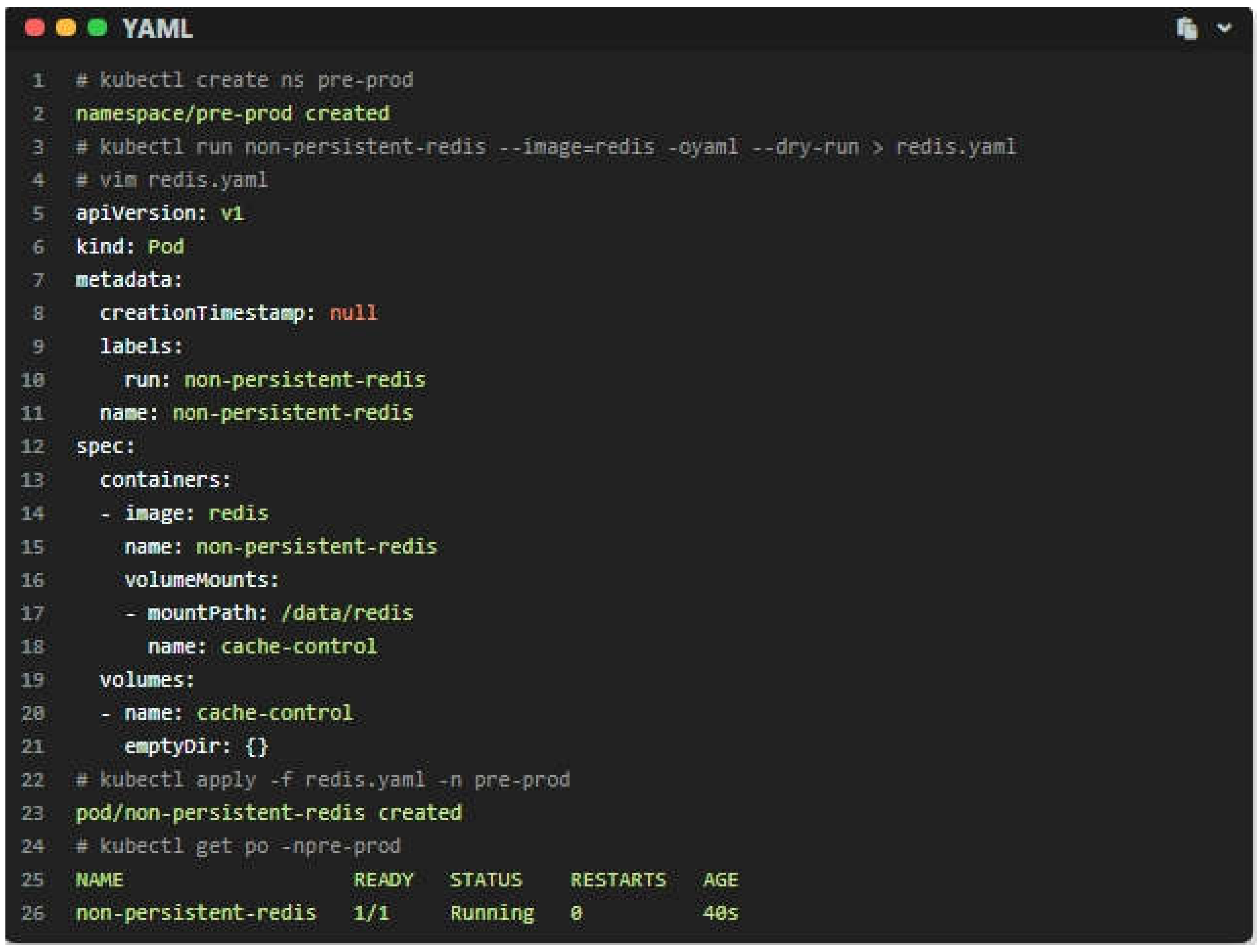
```
# kubectl run pod-secrets-via-env --image=redis -oyaml --dry-run >pod-secrets-via-env.yaml
W0807 17:52:23.764346  104935 helpers.go:535] --dry-run is deprecated and can be replaced with --dry-run=client.
# vim pod-secrets-via-env.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod-secrets-via-env
  name: pod-secrets-via-env
spec:
  containers:
  - image: redis
    name: pod-secrets-via-env
    env:
      - name: TOPSECRET
        valueFrom:
          secretKeyRef:
            name: super-secret
            key: Credential
# kubectl apply -f pod-secrets-via-env.yaml
pod/pod-secrets-via-env created

# kubectl exec -it pod-secrets-via-env bash
root@pod-secrets-via-env:/data# echo $TOPSECRET
alice
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 15
Set configuration context $ kubectl config use-context k8s

Create a pad as follows:
Name: non-persistent-redis
Container image: redis
Named-volume with name: cache-control
Mount path: /data/redis
It should launch in the pre-prod namespace and the volume MUST NOT be persistent.

A. 

```
# kubectl create ns pre-prod
namespace/pre-prod created
# kubectl run non-persistent-redis --image=redis -oyaml --dry-run > redis.yaml
# vim redis.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: non-persistent-redis
  name: non-persistent-redis
spec:
  containers:
  - image: redis
    name: non-persistent-redis
    volumeMounts:
    - mountPath: /data/redis
      name: cache-control
  volumes:
  - name: cache-control
    emptyDir: {}
# kubectl apply -f redis.yaml -n pre-prod
pod/non-persistent-redis created
# kubectl get po -npre-prod
NAME                   READY   STATUS    RESTARTS   AGE
non-persistent-redis   1/1     Running   0          40s
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 16
Set configuration context $ kubectl config use-context k8s

Scale the deployment webserver to 6 pods

A. # kubectl scale deploy/webserver --replicas=6

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 17
Set configuration context $ kubectl config use-context k8s

Check to see how many nodes are ready (not including nodes tainted NoSchedule) and write the number to /opt/nodenum

A.
```
1. # for i in `kubectl get nodes|grep Ready|grep -v 'NAME'|awk '{print $1}'`
2. do
3. kubectl describe node $i |grep Taints|grep -v 'NoSchedule'
4. done
5. Taints:            <none>
6. Taints:            <none>
7. # for i in `kubectl get nodes|grep Ready|grep -v 'NAME'|awk '{print $1}'` ;do kubectl describe node $i |grep Taints|grep -v 'NoSchedule';done|wc -l >/opt/
   nodenum
8. # cat /opt/nodenum
9. 2
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 18
Set configuration context $ kubectl config use-context k8s

From the Pod label name=cpu-utilizer, find pods running high CPU workloads and write the name of the Pod consuming most CPU to the file /opt/cpu.txt (which already exists)

A.
```
1. # for i in `kubectl get po -l k8s-app=kube-dns -nkube-system|grep -v 'NAME'|awk '{print $1}'` ;do kubectl top po $i -nkube-system ;done |sort -k 2r
2. NAME                      CPU(cores)   MEMORY(bytes)
3. NAME                      CPU(cores)   MEMORY(bytes)
4. coredns-7ff77c879f-sbg6j   3m          16Mi
5. coredns-7ff77c879f-pwpgq   3m           12Mi
6.
7. # echo 'coredns-7ff77c879f-sbg6j' >/opt/cpu.txt
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 19
Set configuration context $ kubectl config use-context k8s

Create a deployment as follows

Name: nginx-dns
Exposed via a service: nginx-dns
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 look up the DNS records of the service & pod and write the output to /opt/service.dns and /opt/pod.dns respectively.

Ensure you use the busybox:1.28 image(or earlier) for any testing, an the latest release has an unpstream bug which impacts thd use of nslookup.

A.
```
# kubectl create deploy nginx-ds --image=nginx --image=busybox:1.28 --dry-run -oyaml > nginx-ds.yaml
# vim nginx-ds.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx-ds
  name: nginx-ds
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx-ds
  strategy: {}
  template:
    metadata:
      labels:
        app: nginx-ds
    spec:
      containers:
      - image: nginx
        name: nginx
      - image: busybox:1.28
        name: busybox
        command: ['sh','-c','sleep 3600']

# kubectl apply -f nginx-ds.yaml
deployment.apps/nginx-ds created
# kubectl expose deploy/nginx-ds --port=80

# kubectl exec -it deploy/nginx-ds -c busybox  nslookup nginx-ds>/opt/service.dns
# kubectl exec -it deploy/nginx-ds -c busybox  nslookup nginx-ds >/opt/service.dns
# cat /opt/service.dns
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      nginx-ds
Address 1: 10.98.149.137 nginx-ds.default.svc.cluster.local


# kubectl exec -it deploy/nginx-ds -c busybox  nslookup 10.244.1.101 >/opt/pod.dns
# cat /opt/pod.dns
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.244.1.101
Address 1: 10.244.1.101 nginx-ds-576c7d4d77-m47rt
```

Correct Answer: A
Explanation

Explanation/Reference:

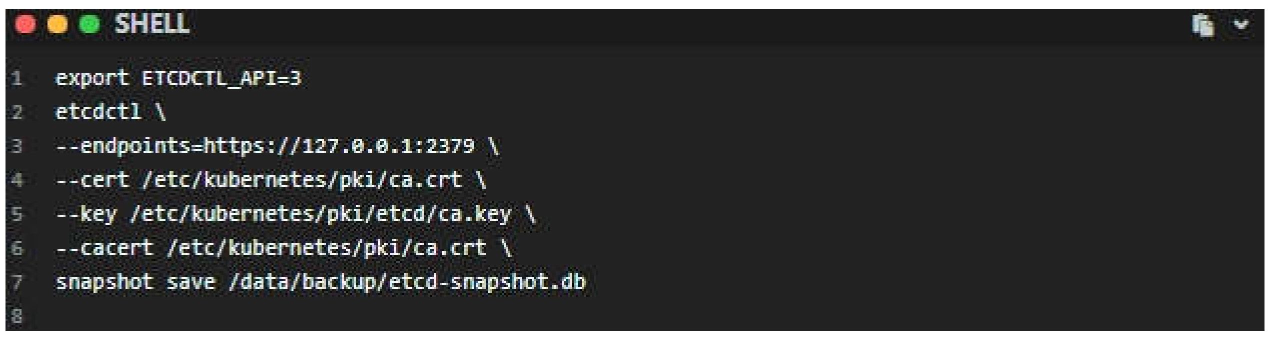
QUESTION 20
No configuration context change required for this item

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

The etcd instance is running etcd version 3.1.10

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
Clientkey:/opt/KUCM00302/etcd-client.key

A.


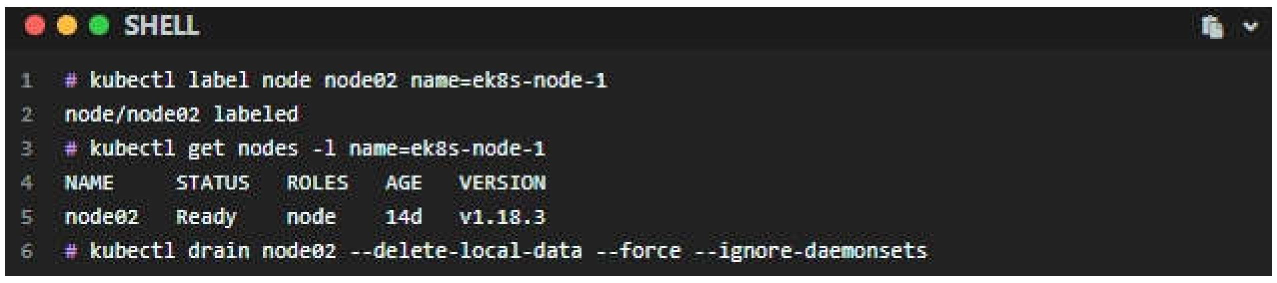
```
export ETCDCTL_API=3
etcdctl \
--endpoints=https://127.0.0.1:2379 \
--cert /etc/kubernetes/pki/ca.crt \
--key /etc/kubernetes/pki/etcd/ca.key \
--cacert /etc/kubernetes/pki/ca.crt \
snapshot save /data/backup/etcd-snapshot.db
```

Correct Answer: A
Explanation

Explanation/Reference:
https://kubernetes.io/zh/docs/tasks/administer-cluster/configure-upgrade-etcd/#%E5%A4%87%E4%BB%BD-etcd-%E9%9B%86%E7%BE%A4

QUESTION 21
Set configuration context $ kubectl config use-context ek8s

Set the node labelled with name=ek8s-node-1 as unavailable and reschedule all the pods running on it.

A. 

```
# kubectl label node node02 name=ek8s-node-1
node/node02 labeled
# kubectl get nodes -l name=ek8s-node-1
NAME     STATUS   ROLES   AGE   VERSION
node02   Ready    node    14d   v1.18.3
# kubectl drain node02 --delete-local-data --force --ignore-daemonsets
```

Correct Answer: A
Explanation

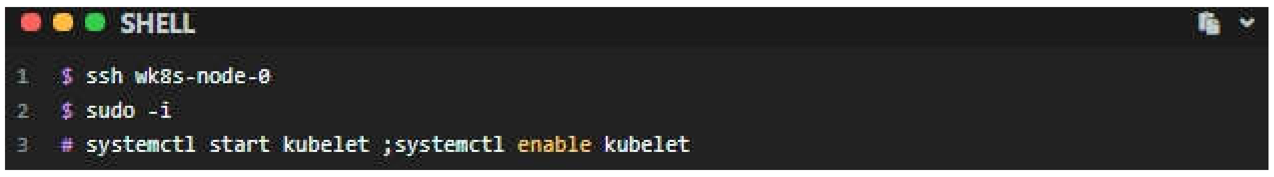
Explanation/Reference:

QUESTION 22
Set configuration context $ kubectl config use-context wk8s

A Kubernetes worker node, labelled with name=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.

Hints:

You can ssh to the failed node using $ ssh wk8s-node-0
You can assume elevated privileges on the node with the following command $ sudo -i

A. 

```
$ ssh wk8s-node-0
$ sudo -i
# systemctl start kubelet ;systemctl enable kubelet
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 23
Set configuration context $ kubectl config use-context wk8s

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 nginx named myservice automatically. Any spec files required should be placed in the /etc/kubernetes/manifests directory on the node.

Hints:

1. You can ssh to the failed node using $ ssh wk8s-node-1
2. You can assume elevated privileges on the node with the following command $ sudo -i

A.
```
1.  $ ssh wk8s-node-1
2.  $ sudo -i
3.  # vim /etc/kubernetes/manifests/myservice.yaml
4.  apiVersion: v1
5.  kind: Pod
6.  metadata:
7.    creationTimestamp: null
8.    labels:
9.      run: myservice
10.   name: myservice
11. spec:
12.   containers:
13.   - image: nginx
14.     name: myservice
15.     resources: {}
16.   dnsPolicy: ClusterFirst
17.   restartPolicy: Always
18. status: {}
19. # systemctl status kubelet
20. ● kubelet.service - kubelet: The Kubernetes Node Agent
21.    Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: disabled)
22.   Drop-In: /usr/lib/systemd/system/kubelet.service.d
23.            └─10-kubeadm.conf
24. # vim /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf
25. # Note: This dropin only works with kubeadm and kubelet v1.11+
26. [Service]
27.   vironment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig
28. =/etc/kubernetes/kubelet.conf"
29. Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
30. # vim /var/lib/kubelet/config.yaml
31. staticPodPath: /etc/kubernetes/manifests
32. # systemctl restart kubelet
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 24
Set configuration context $ kubectl config use-context bk8s

Given a partially-functioning Kubenetes cluster, identify symptoms of failure on the cluster. Determine the node, the failing service and take actions to bring up the failed service and restore the health of the cluster. Ensure that any changes are made permanently.

The worker node in this cluster is labelled with name=bk8s-node-0 Hints:

You can ssh to the relevant nodes using $ ssh $(NODE) where $(NODE) is one of bk8s-master-0 or bk8s-node-0
You can assume elevated privileges on any node in the cluster with the following command$ sudo -i

A.
```
# cd /etc/kubernetes/manifests/
# ls
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
# vim /var/lib/kubelet/config.yaml
staticPodPath: /etc/kubernetes/DODKSIYF => /etc/kubernetes/manifests
# systemctl restart kubelet
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 25
Set configuration context $ kubectl config use-context hk8s

Creae a persistent volume with name app-config of capacity 1Gi and access mode ReadWriteOnce. The type of volume is hostPath and its location is /srv/app-config

A.
```
# vim app-config.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-config
  labels:
    type: local
spec:
  storageClassName: app-config
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/srv/app-config"


# kubectl apply -f app-config.yaml
persistentvolume/app-config created
# kubectl get pv
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
app-config   1Gi        RWO            Retain           Available           app-config              4s
```

Correct Answer: A
Explanation

Explanation/Reference:

QUESTION 26
Containers are run on which of these?
Choose the Answer:

A. Services
B. Controllers
C. Nodes
D. None of these

Correct Answer: C
Explanation

Explanation/Reference:
Nodes run the pods.

QUESTION 27
Kubernetes changed the name of cluster members to "Nodes." What were they called before that? Choose the Answer:

A. Workers
B. Cogs
C. Minions
D. Slaves

Correct Answer: C
Explanation

Explanation/Reference:
A lot of documentation and tutorials online still refer to worker nodes this way.

QUESTION 28
Unique IP addresses are assigned to:
Choose the Answer:

A. NAT is used extensively, so unique IP addresses are irrelevant
B. Pods
C. Container Hosts
D. Containers

Correct Answer: B
Explanation

Explanation/Reference:
A pod gets assigned a single IP address, regardless of how many containers make it up. This is analogous to many services running on a single virtual machine.

QUESTION 29
Usually, when submitting a Kubernetes API call, data is sent in which format? (Select all that apply)
Choose the 2 correct answers:

A. YAML
B. XML
C. DOC
D. JSON

Correct Answer: AD
Explanation

Explanation/Reference:
If using a direct API call in an application, JSON is used. If using kubectl to submit a request, it takes YAML.

QUESTION 30
Which of these are not inherently created by Kubernetes? Choose the Answer:

A. Services
B. Nodes
C. Controllers
D. Pods

Correct Answer: B
Explanation

Explanation/Reference:
Nodes are added to a cluster, and a Kubernetes object is created to reflect them, but Kubernetes itself doesn't create them.

QUESTION 31
Communications between the apiserver and the kubelet on the cluster nodes are used for all but which of the following?
Choose the Answer:

A. Providing the kubelet's port-forwarding capability
B. Fetching logs for pods
C. Keep-alive xml packets
D. Attaching (through kubectl) to running pods

Correct Answer: C
Explanation

Explanation/Reference:
Communications between the apiServer and the Kubelet are constantly communicating for a variety of purposes.

QUESTION 32
The connection between the apiserver and nodes, pods and services:
Choose the Answer:

A. Is unencrypted and therefore unsafe to run over public networks.
B. Is always encrypted with IPSec.
C. Is always encrypted using the method configured in the .kube file.
D. Is currently encrypted with IPSec with plans to allow other encryption plugins later.

Correct Answer: A
Explanation

Explanation/Reference:
It's a fairly simple process to encrypt the streams using TLS.

QUESTION 33
If memory is running low on a running node, which of these keys will return "True"? Choose the Answer:

A. OOM
B. Warning
C. MemoryPressure
D. LowMemory

Correct Answer: C
Explanation

Explanation/Reference:
MemoryPressure and DiskPressure return true as a node starts to become overcommitted.

QUESTION 34
What does a pod represent in a Kubernetes cluster?
Choose the Answer:

A. A running process
B. Conditions under which applications will autoscale
C. A set of rules for maintaining high availability
D. All the containers in the cluster

Correct Answer: A
Explanation

Explanation/Reference:
Pods are the running containers in a Kubernetes cluster.

QUESTION 35
Which of these components mount volumes to containers? Choose the Answer:

A. kube-proxy
B. fluentd
C. kubelet
D. kube-scheduler

Correct Answer: C
Explanation

Explanation/Reference:
The kubelet which runs on nodes handles moment-to-moment management of the pods on its node.

QUESTION 36
What is the difference between a Docker volume and a Kubernetes volume? Choose the Answer:

A. Proximity: In Docker, volumes can reside on the same host with their containers. In Kubernetes, they must reside on separate metal for resiliency.
B. Back-end Drivers. Docker supports more block storage types than Kubernetes does.
C. Size: Docker volumes are limited to 3TB. Kubernetes volumes are limited to 16TB.
D. Volume lifetimes. In Docker, this is loosely defined. In Kubernetes, the volume has the same lifetime as its surrounding pod.

Correct Answer: D
Explanation

Explanation/Reference:
Docker volumes are not used in conjunction with Kubernetes at this time.

QUESTION 37
In a typical deployment, the Kubernetes Master listens on what port number? Choose the Answer:

A. 22
B. 3001
C. 80
D. 443

Correct Answer: D
Explanation

Explanation/Reference:
The API server, by default, listens on port 443, the secure HTTP port.

QUESTION 38
In Kubernetes, a group of one or more containers is called:
Choose the Answer:

A. A selector
B. A pod
C. A swarm
D. A minion

Correct Answer: B
Explanation

Explanation/Reference:
A pod is usually one container but can be a group of containers working together.

QUESTION 39
Which of these is a list of common Kubernetes primitives? Choose the Answer:

A. service, deployment, replicaset, etcd
B. pod, service, persistentVolume, deployment
C. containers, vms, hypervisors, daemons
D. pod, swarm, namespace, network

Correct Answer: B
Explanation

Explanation/Reference:
There are many others, but those are the ones you'll likely work with most often.

QUESTION 40
What controls a Kubernetes cluster?
Choose the Answer:

A. minikube
B. The Master
C. kube-proxy
D. kubelet

Correct Answer: B
Explanation

Explanation/Reference:
The master node contains the Kubernetes api server, which controls what the cluster does.

QUESTION 41
For network policies to work in Kubernetes, which of these must be true? Choose the Answer:

A. The CNI must have a "policy" sidebar.
B. The CNI must support VxLANs.
C. Network policies are always enforced.
D. The CNI must enforce the network policies.

Correct Answer: D
Explanation

Explanation/Reference:
If the CNI doesn't support network policies, then applying a YAML formula with a network policy in it will return a success, but the policies will not be enforced.

QUESTION 42
Which platform(s) will Minikube run on? (Select all that apply) Choose the 3 correct answers:

A. Mac OS X
B. Windows
C. Novell Netware v4
D. Linux

Correct Answer: ABD
Explanation

Explanation/Reference:
And probably, just for spite, someone will port it to Novell Netware so we'll have to change this question, but Minikube should run just about anywhere.

QUESTION 43
To deploy Kubernetes using kubeadm, you'll have to choose:
Choose the Answer:

A. The amount of RAM allocated to the Kubelets
B. An appropriate CNI (Container Network Interface)
C. Between container space and swap space
D. A passphrase for the certificates

Correct Answer: B
Explanation

Explanation/Reference:
kubeadm doesn't make any provisions for inter-node networking. There are a lot of CNIs to choose from!

QUESTION 44
What is the node called that runs the api server?
Choose the Answer:

A. The Server
B. The Top
C. The Client
D. The Master

Correct Answer: D
Explanation

Explanation/Reference:
The Master node runs the api server and is where Kubernetes accepts requests via a RESTful API.

QUESTION 45
What do many Kubernetes deployment tools handle automatically for you? Choose the Answer:

A. Kubectl installation on the master and nodes
B. Custom namespaces
C. Certificate creation
D. CNI deployment

Correct Answer: C
Explanation

Explanation/Reference:
Most deployment tools handle the certificate creation but will not do the other things.

QUESTION 46
Which of these is not a CNI provider?
Choose the Answer:

A. Canal
B. Flannel
C. Weave Net
D. Ceph

Correct Answer: D
Explanation

Explanation/Reference:
Ceph is an object store, the other three are CNI providers.

QUESTION 47
Which of these is an inexpensive and easy way to try out Kubernetes? Choose the Answer:

A. Manual Install
B. Linux Foundation's CNI
C. Turnkey
D. Minikube

Correct Answer: D
Explanation

Explanation/Reference:
Minikube is a great and inexpensive way to try out Kubernetes.

QUESTION 48
In Kubernetes, one of the primitives is a Node (which was formerly referred to as a "Minion").
What does it represent?
Choose the Answer:

A. A virtual machine running the Kubelet and doing the compute work via Docker.
B. A physical machine running the Kubelet and doing the compute work via a container service like Docker or Rocket.
C. A physical or virtual machine running the Kubelet and doing the compute work via a container service like Docker or Rocket.
D. A virtual machine running the Kubelet and doing the compute work via a container service like Docker or Rocket.

Correct Answer: C
Explanation

Explanation/Reference:
While nodes are generally considered to be physical machines, as that's the norm in production deployments, they can be virtual machines as well.

QUESTION 49
What is the default encryption used in Kubernetes? (Choose the answer that is most correct.) Choose the Answer:

A. SSL
B. TLS
C. AES
D. HTTPS

Correct Answer: B
Explanation

Explanation/Reference:
TLS is the default encryption used in Kubernetes.

QUESTION 50
What underlying technology does Flannel use to allow pods to communicate? Choose the Answer:

A. GRE Tunnels
B. VLANs
C. IPSec Tunnels
D. VxLANs

Correct Answer: D
Explanation

Explanation/Reference:
Flannel uses VxLANs for the overlay network among the pods.

QUESTION 51
How is authorization handled in Kubernetes?
Choose the Answer:

A. A built-in Role Based Access Control system.
B. Through a variety of third-party authorization plugins.
C. LDAP/AD
D. Through user.permission files mounted via secrets

Correct Answer: A
Explanation

Explanation/Reference:
K8s has its own RBAC components built it.

QUESTION 52
Which types of API requests should be authenticated? Choose the Answer:

A. Incoming requests from proxies
B. Requests from users
C. Node requests
D. All of them

Correct Answer: D
Explanation

Explanation/Reference:
Everything, every time. Don't allow security holes in your cluster!

QUESTION 53
You are writing YAML for a pod, and want to limit its CPU utilization to one quarter of the CPU. Which of the following lines will most likely be in your final YAML file? (Ignore whitespace) Choose the Answer:

A. cpu: "0.250m"
B. cpu: "1:4"
C. cpu: "25"
D. cpu: "250m"

Correct Answer: D
Explanation

Explanation/Reference:
250m stands for 250 millicpus, which works out to 1/4 of a running CPU.

QUESTION 54
Is it possible to configure an application in a container from Kubernetes? If so, how is this accomplished?
Choose the Answer:

A. Yes, through the use of environment variables. These can be set in the YAML file for the appropriate pod.
B. Yes, through the use of annotations. Annotations are key/value pairs used by the applications in the service.
C. No, this is not possible at this time but is planned for a future release.
D. Yes, through the use of Network Policies. While originally intended to be the traffic cops of the network, developers often use them "off label" to configure applications.

Correct Answer: A
Explanation

Explanation/Reference:
Environment variables all the way. These get set up in the YAML file and passed through to the container so that applications running inside have access to the relevant information.

QUESTION 55
There are many ways to assign a pod to a particular node, but they all involve the use of what? Choose the Answer:

A. affinity or anti-affinity
B. kubectl
C. labels
D. annotations

Correct Answer: C
Explanation

Explanation/Reference:
They all use labels. Kubectl was a red herring. Remember, you *could* do this using Kubernetes API calls and not use kubectl at all. :)

QUESTION 56
Which of these commands would scale up a deployment called "soup" from 3 pods to 5? Choose the Answer:

A. kubectl scale current-replicas=3 replicas=5 ds/soup
B. kubectl scale current-replicas 3 replicas 5 ds soup
C. kubectl scale replicas=3 soup
D. kubectl scale replicas=5 deployment/soup

Correct Answer: D
Explanation

Explanation/Reference:
"ds" is the short form for DaemonSets, not deployments! You don't *have* to use the current-replicas flag, and if you do, remember that it will *only* scale up the deployment *if* the current number of replicas matches that number.

QUESTION 57
Which of these is the correct hierarchy of related Kubernetes API objects? Choose the Answer:

A. Pods run services, which in turn are managed by deployments.
B. Services point to pods. Pods are made up of deployments.
C. Pods, services, and deployments refer to the same level of hierarchy in K8s.
D. Pods make up deployments. Services point to deployments.

Correct Answer: D
Explanation

Explanation/Reference:
Pods are the simplest Kubernetes API object. Deployments manage pods. Services point to deployments.

QUESTION 58
What are labels used for?
Choose the Answer:

A. Selecting objects for a variety of purposes.
B. Setting environment variables in the container on a pod.
C. Human-readable descriptions of objects. They have no other use.
D. Setting the image version number on a container in a pod. They have no other use.

Correct Answer: A
Explanation

Explanation/Reference:
Labels are incredibly useful tools! They can be used to select pods for networking policies, select all the pods serving a particular app, or any other way you might need to group your pods together. Careful and thoughtful application of labels makes managing large deployments easy.

QUESTION 59
Which parameter is used to increase or decrease the number of pods that make up a deployment?
Choose the Answer:

A. Syncs
B. Nodes
C. Replicants
D. Replicas

Correct Answer: D
Explanation

Explanation/Reference:
The number of replicas tells K8s how many pods to keep running at all times. It's easy to scale applications up and down using replicas.

QUESTION 60
Which of these is the best use case for a DaemonSet? Choose the Answer:

A. A monitoring back-end that only needs intermittent network access.
B. A CNI container that needs to run on every node in order to function properly.
C. A stateless web-head that will be load-balanced among many nodes.
D. A MariaDB/Galera cluster that must autoscale depending on CPU utilization.

Correct Answer: B
Explanation

Explanation/Reference:
DaemonSets are most useful for deploying pods on every node (or selecting specific nodes to run the pods on).

QUESTION 61
Which of these is a difference between annotations and labels in Kubernetes? Choose the Answer:

A. Labels are used to select and identify objects. Annotations are not.
B. They are the same thing.
C. Labels allow a wider variety of characters to be used in their names than annotations.
D. Annotations use a key/value pair config map.

Correct Answer: A
Explanation

Explanation/Reference:
Both use key/value pair config maps, and annotations allow for a wider variety of characters that labels do not allow.

QUESTION 62
I have a deployment called "healer" running on my cluster. I look at the pods on a node and notice that there are two pods running there "healer-xxxxxxxx-yyyy" and "healer-xxxxxxxx-yyyz". What will happen if I issue the command "kubectl delete pod healer-xxxxxxxx-yyyz"?
Choose the Answer:

A. Nothing. The pod is protected by the deployment it runs in.
B. The pod will be deleted, but the deployment will immediately spin up another pod to replace it, possibly on another node.
C. The pod will be deleted. If there is an Ingress or Service Load balancer pointing to that pod, those requests will time out.
D. Kubectl will issue an error message, as this pod is in use. Adding the force flag will allow you to complete this action.

Correct Answer: B
Explanation

Explanation/Reference:
The power of Kubernetes is that it self-heals, even if the administrator is unknowingly (or knowingly) taking down pods in a deployment.

QUESTION 63
What is the scheduler?
Choose the Answer:

A. An isolated, non-containerized process on the master node.
B. A subprocess of the CNI.
C. A pod on the master node.
D. A distributed DaemonSet on the cluster.

Correct Answer: C
Explanation

Explanation/Reference:
The scheduler is a process that runs in a pod, usually on the master node. While it's unusual, it's possible to have multiple schedulers running on the same cluster.

QUESTION 64
If a pod requests more resources than is available on any given node, what happens? Choose the Answer:

A. The pod will move into a "NotReady" status.
B. The scheduler will return an error.
C. The pod will not be scheduled until a node with the resources becomes available.
D. The pod will get scheduled on the master node.

Correct Answer: C
Explanation

Explanation/Reference:
The pod will remain in a "Pending" status until a node becomes available which might be never.

QUESTION 65
Why might a user desire two pods to have anti-affinity? Choose the Answer:

A. She wants them to run on different nodes to avoid sharing failure domains.
B. She wants them to run on the same node to speed up networking traffic between them.
C. She wants them to be on network adjacent nodes for faster shared disk access.
D. She wants them to share memory space on a node.

Correct Answer: A
Explanation

Explanation/Reference:
Anti-affinity means that two pods will not run on the same node, and is usually implemented to prevent two pods from being in the same failure domain in case something goes wrong.

QUESTION 66
Why are annotations particularly important when using multiple or custom schedulers? Choose the Answer:

A. Because multiple schedulers are not allowed without annotations because of the security risk.
B. Because they are the only audit trail available for the scheduler.
C. Because they are how the scheduler is specified.
D. Because they can remind operators which scheduler was used to place (or fail to place) a pod.

Correct Answer: D
Explanation

Explanation/Reference:
Annotations are designed to provide additional non-identifying information about a pod, and things like application version or scheduler that placed the pod are ideal uses for these.

QUESTION 67
If a toleration and a taint match during scheduling, what happens? Choose the Answer:

A. The toleration is ignored and the node might be scheduled for uncordon.
B. The taint is ignored and the pod might be scheduled to the node.
C. The toleration and taint reinforce one another, further guaranteeing that the pod is not scheduled on the node.
D. An error taints and tolerations cannot be used together in the same namespace.

Correct Answer: B
Explanation

Explanation/Reference:
Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints. Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

QUESTION 68
How can a user specify which scheduler a pod should use? Choose the Answer:

A. Through the schedulerName tag in the spec.
B. By adding a schedule=custom label to the metadata.
C. Through the scheduler-name tag in the spec.
D. By adding a schedulerName=*scheduler* annotation to the metadata.

Correct Answer: A
Explanation

Explanation/Reference:
The tag for specifying a particular scheduler is schedulerName and defaults to default-scheduler.

QUESTION 69
What are taints and what are they applied to?
Choose the Answer:

A. Taints are used to repel certain pods from nodes and are applied to nodes.
B. Taints are used to repel workloads with certain labels and are applied to nodes and pods.
C. Taints are used to mark a pod as unavailable during an outage and are applied to pods.
D. Taints are used to repel workloads from one another (anti-affinity) and are applied to pods.

Correct Answer: A
Explanation

Explanation/Reference:
Taints allow a node to repel a set of pods.

QUESTION 70
When an API request is made to create a pod, which piece determines which node will be used to instantiate the new pod?
Choose the Answer:

A. The API Server itself
B. The Kubelet on the target node
C. The scheduler
D. kube-proxy finds a free node

Correct Answer: C
Explanation

Explanation/Reference:
The scheduler is what determines which pods go with which nodes.

QUESTION 71
What is podAffinity used for?
Choose the Answer:

A. Ensuring replicated pods in the same deployment are placed on different nodes.
B. Preventing two pods from being placed on the same node.
C. Allowing nodes with containers in the same pod access to a higher-speed network.
D. Placing two or more pods on the same node.

Correct Answer: D
Explanation

Explanation/Reference:
Placing two or more pods on the same node is done with the podAffinity attribute and uses labels.

QUESTION 72
How can a pod be assigned to a specific node?
Choose the Answer:

A. Using a nodeSelector with properly labelled nodes.
B. The scheduler does not allow for pods to be placed manually.
C. Set node constraints in the node's YAML.
D. Use the host property in the pod's YAML.

Correct Answer: A
Explanation

Explanation/Reference:
Using the nodeSelector is the easiest way to manually assign pods to nodes.

QUESTION 73
What's the recommended method for dealing with applications that insist on writing out logs to a file rather than being able to redirect them to stdout? Choose the Answer:

A. Find a logging agent that can operate inside the pod and send the logs to a central file store or log aggregator.
B. Don't use Kubernetes.
C. Do without logging.
D. Redirect the log file to ephemeral storage on the host.

Correct Answer: A
Explanation

Explanation/Reference:
It's usually a fairly easy task to incorporate a logging handler and central location for log files within the cluster.

QUESTION 74
Which command will give you stdout of a pod called "to-the-screen"? Choose the Answer:

A. kubectl exec -it to-the-screen kube-get-stdout
B. kubectl get logs po to-the-screen
C. kubectl logs -f to-the-screen.yaml
D. kubectl logs to-the-screen

Correct Answer: D
Explanation

Explanation/Reference:
kube logs is the fastest way to get stdout and the recommended, standard way to configure your applications in containers to handle their logs.

QUESTION 75
I have a node called "node8" and I'd like to know what kind of load it's under including the CPU and Memory requests. Which of these commands would give me that information? Choose the Answer:

A. kubectl get nodes status{cpu.requests}&&{memory.requests}
B. kubectl describe deployments all-namespaces with-nodes
C. kubectl describe node node8
D. kubeadm status node8

Correct Answer: C

Explanation

Explanation/Reference:
kubectl describe node will give you all kinds of very useful up to date information about a given node.

QUESTION 76
Where does the Kubernetes key/value store (etcd) log file reside? Choose the Answer:

A. On the host in /var/log/kubernetes/etcd
B. On the host in /var/log/pods
C. On the host in /var/syslog
D. On the host in /etc/kubernetes/etcd.log

Correct Answer: B
Explanation

Explanation/Reference:
The Kubernetes services that run in pods on the host store their logs in /var/log/pods

QUESTION 77
Is it possible to get a shell prompt to a Ubuntu 16.04 based container called "sidecar1" in the pod "star-aaaaaaaaaa-bbbbb"? There are several containers in the pod. If so, how? Choose the Answer:

A. Yes! kubectl exec -it star-aaaaaaaaaa-bbbbb container sidecar1 /bin/bash
B. Yes! kubectl run star-aaaaaaaaaa-bbbbb sidecar1 /bin/bash
C. No. This is only possible when there is a single container in the pod.
D. Yes! kubectl exec -it star-aaaaaaaaaa-bbbbb/sidecar1 /bin/bash

Correct Answer: A
Explanation

Explanation/Reference:
While it's discouraged in Kubernetes, it's still possible to get to a container's shell. It's generally considered a bad idea to do things like alter configuration files or apt-get files while logged in. Its use should be limited to debugging when possible.

QUESTION 78
What's an easy command to check the health and status of your cluster? Choose the Answer:

A. kubeadm k8s-status
B. kubectl create -f status
C. kubectl cluster-status
D. kubectl get nodes

Correct Answer: D
Explanation

Explanation/Reference:
Kubectl get nods will show you at a glance which of your nodes are ready and which might be having troubles. It's a great first stop if you suspect trouble.

QUESTION 79
Which log command will show you just the final 8 lines of stdout for a pod? Choose the Answer:

A. kubectl logs tail=8
B. kubectl get logs tail=8
C. kubectl logs tail 8

Correct Answer: A
Explanation

Explanation/Reference:
Two hyphens and an equal, unless you want exactly ten lines, then it's just kubectl logs tail

QUESTION 80
Is it possible to get the logs back from a dead or evicted pod? If so, how? Choose the Answer:

A. No, once a pod is gone, all of its ephemeral storage is gone.
B. Yes, if the node is immediately cordoned, you can use the inspect flag.
C. Yes, add the previous flag to the kubectl logs command.
D. Yes, restart the dead pod in safe mode and extract the file through scp or sftp.

Correct Answer: C
Explanation

Explanation/Reference:
To grab the last logs, just add previous!

QUESTION 81
Starting with Kubernetes 1.8, there is a new metrics API. This can be accessed directly from the command line with which command?
Choose the Answer:

A. cadvisor list
B. heapster get info
C. kubectl metrics [nodes | pods]
D. kubectl top [nodes | pods]

Correct Answer: D
Explanation

Explanation/Reference:
kubectl top, along with the object you'd like to watch, gives some in-depth information right on the command line. Who needs a GUI?

QUESTION 82
I'm troubleshooting an application issue and would love to see the application's logs, which are in a file in the container "appctn" in the pod "apppod-abcdef123-45678" at /var/log/applog.
Which of these commands would list that for me?
Choose the Answer:

A. kubectl logs -c appctn
B. kubectl exec apppod-abcdef123-45678 cat /var/log/applog
C. kubectl logs apppod-abcdef123-45678 -c appctn
D. kubectl logs apppod-abcdef123-45678

Correct Answer: B
Explanation

Explanation/Reference:
kubectl logs only work for STDOUT, so if your logs are elsewhere, you'll need to pull them with something like the command here.

QUESTION 83
You need to install a brand new network card into a node in your cluster, called "slowpoke". How do you prepare Kubernetes for the shutdown and outage without affecting the rest of your cluster? One of these commands will keep new pods from being scheduled to the node during the outage. Once the node is back up and running, the other command will allow scheduling to resume. SELECT TWO.
Choose the 2 correct answers:

A. kubectl uncordon slowpoke
B. kubectl drain slowpoke
C. kubectl fill slowpoke
D. kubectl cordon slowpoke

Correct Answer: AB
Explanation

Explanation/Reference:
drain and uncordon are the commands.

QUESTION 84
If you want to easily add nodes to your cluster, what mode should it be in? Choose the Answer:

A. Auto-Add Mode
B. Node Toad Code Road Mode
C. Master Scan Mode
D. Node Self Registration Mode

Correct Answer: D
Explanation

Explanation/Reference:
When the kubelet flag register-node is true (the default), the kubelet will attempt to register itself with the API server. This is the preferred pattern, used by most distros.

QUESTION 85
Your colleague is responsible for a Kubernetes cluster and wants to be on the latest version but also wants to disallow any features that are not part of the v1 API. Can this be done? If so, how? Choose the Answer:

A. The features and the version are the same. If he wants to use the v1 API, he needs to install v1.
B. Yes, but it requires configuring each kubelet on each node with runtime-config=api/all=false,api/v1=true
C. No. You can turn off all beta and/or legacy functionality but cannot specify a particular API version.
D. Yes. He just needs to append runtime-config=api/all=false,api/v1=true to the command that brings up his apiserver.

Correct Answer: D
Explanation

Explanation/Reference:
The apiserver is the only piece that requires configuration to make this possible.

QUESTION 86
Ordinarily, you should only use kubectl drain on one node at a time. What happens if you attempt to drain more nodes in parallel?
Choose the Answer:

A. The drain is always allowed, which could put your application states in jeopardy, so use with caution!
B. As long as the over-commit threshold has not been reached on the remaining nodes, the drain is allowed.
C. The second and subsequent drains are blocked until the drained node is uncordoned.
D. Any drains that would cause the number of ready replicas to fall below the specified budget are blocked.

Correct Answer: D
Explanation

Explanation/Reference:
The Stateful Sets take care of themselves and prevent drains from happening that would prevent them from maintaining their budgets.

QUESTION 87
You've used kubeadm to upgrade your cluster from Kubernetes 1.8 to 1.9. Several of the nodes failed because of an accidental shutdown during the upgrade. All nodes are running now, some in the v1.8 state, some in v1.9. What's your next step? Choose the Answer:

A. 1.8 and 1.9 should work fine together, so it's safe to leave them as they are.
B. Shut down the nodes that have successfully upgraded run kubectl upgrade again.
C. Run kubeadm upgrade again. It is idempotent and will move all nodes to the desired state.

D. There isn't much to do to fix this except manually upgrade each of the failed nodes to v1.9 and rejoin them to the cluster.

Correct Answer: C
Explanation

Explanation/Reference:
kubectl upgrade on v1.9 will do absolutely nothing to a node that's already been successfully upgraded.

QUESTION 88
If an Ingress request is made with no associated rules, what happens? Choose the Answer:

A. All traffic is forbidden in the namespace.
B. The request fails and no Ingress is created. Rules are required.
C. All traffic is forbidden in the namespace except to the named host.
D. All traffic is sent to a single host.

Correct Answer: D
Explanation

Explanation/Reference:
This is a useful way of setting up common error pages, such as the location of a unified 404 page.

QUESTION 89
What handles inter-pod communication?
Choose the Answer:

A. GRE tunnels
B. The CNI
C. Host networking
D. VLANs

Correct Answer: B
Explanation

Explanation/Reference:
The CNI (Container Network Interface) allows pods to communicate with one another within a cluster regardless of which node they are on.

QUESTION 90
If a service called "web-head" is exposed in the default namespace, then other pods can resolve it using all of these hostnames except which?
Choose the Answer:

A. web-head.local
B. web-head.default
C. web-head
D. All of these will resolve properly.

Correct Answer: A
Explanation

Explanation/Reference:
The .local won't work!

QUESTION 91
When a service type of "ClusterIP" is used, what is the result? Choose the Answer:

A. An IP address in a specialized bridge network that links the external network to the internal cluster network.
B. A port on the node where the pod resides, usually above 30000.
C. A single IP address within the cluster that redirects traffic to a pod (possibly on a different node) serving the application (the pod).
D. An single IP address external to the cluster that is drawn from a pool of available public addresses.

Correct Answer: C
Explanation

Explanation/Reference:
ClusterIP is most commonly used with third-party load balancers.

QUESTION 92
Think about the YAML for a network policy. If you had to create one, what is the pattern? Choose the Answer:

A. Preamble, ingress rules, host(s), egress rules, host(s)
B. Preamble, podSelector, hosts, ingress rules, egress rules
C. Preamble, host, podSelector, ingress, and/or egress rules
D. Preamble, podSelector, ingress, and/or egress rules

Correct Answer: D
Explanation

Explanation/Reference:
Preamble contains apiVersion, Kind, and Metadata; then comes the podSelector to determine which pods this policy oversees; and, finally, the rules.

QUESTION 93
What determines how a set of pods are allowed communicate with one another and other network endpoints?
Choose the Answer:

A. Ingress
B. RBACs
C. PVCs
D. Network Policies

Correct Answer: D
Explanation

Explanation/Reference:
Network policies determine what traffic gets into and out of a pod. The CNI must support them, though, but most of them do.

QUESTION 94
Ingress is fairly new to the Kubernetes stack. What version number was the first one to include it?
Choose the Answer:

A. 1.5
B. 1.1
C. 1.0
D. 1.8

Correct Answer: B
Explanation

Explanation/Reference:
v1.1 of Kubernetes included the Ingress API object and it's been constantly improved and increasingly used ever since.

QUESTION 95
What is required to specify a service type of "LoadBalancer"? Choose the Answer:

A. Three or more pods in a deployment.
B. A cloud provider that supports Kubernetes-provisioned load balancers.
C. A pod to check the health of the other pods.
D. Nothing it's built in.

Correct Answer: B
Explanation

Explanation/Reference:
The "LoadBalancer" service type only works on cloud providers that support it. Minikube will also allow it but does not create a full, production-quality load balancer.

QUESTION 96
For a user to be able to request an Ingress resource, what must the cluster have? Choose the Answer:

A. An Ingress controller compatible with available and appropriate service providers like load balancers.
B. A DaemonSet of redis for storing configuration information.
C. A CNI that supports Ingress.
D. An iSCSI volume to store configuration information.

Correct Answer: A
Explanation

Explanation/Reference:
With Kubernetes, the general rule of thumb is that YAML requests will return successfully, but if there is no service to fulfill it then the request will have no effect.

QUESTION 97
What is an Ingress as it relates to Kubernetes?
Choose the Answer:

A. A port on the master where containers are mapped to pods.
B. An API object that manages external access to the services in a cluster, usually HTTP.
C. An API object that creates a services load balancer to access services in the cluster from alternate nodes.
D. A method of routing control-plane instructions to the master node.

Correct Answer: B
Explanation

Explanation/Reference:
A fairly new concept in Kubernetes, an Ingress allows us to abstract away the implementation details of routes into the cluster, such as Load Balancers.

QUESTION 98
Which of the following is true about a volume?
Choose the Answer:

A. The storage class argument is used to specify a PersistentVolumeClaim's requested access mode.
B. Volumes must be statically allocated by the administrator before a user can request one.
C. Volumes must reside on a cluster host.
D. Regardless of which access modes a volume supports, it can only support one at a time.

Correct Answer: D
Explanation

Explanation/Reference:
For example, if a volume is mounted as ReadOnlyMany in one pod, then another pod may not mount it in ReadWriteMany, even if it is supported by the volume type.

QUESTION 99
What are the access modes that can be requested for a volume? Choose the Answer:

A. ReadWriteOnce, ReadOnlyMany, ReadWriteMany
B. Read, Write, ConcurrentWrite
C. ReadOnly, ReadWrite, ReadWriteMany
D. Single, Multi, ROX

Correct Answer: A
Explanation

Explanation/Reference:
Access modes are in terms of how many pods can read or write to a volume simultaneously.

QUESTION 100
PersistentVolumes (PVs) are cluster storage resources. How do they access different kinds of physical storage from different vendors?
Choose the Answer:

A. Kubernetes only supports ephemeral storage
B. Local node storage available as host filesystems
C. Plug-ins
D. NAS supporting the KNS protocol

Correct Answer: C
Explanation

Explanation/Reference:
There are many different plug-ins in Kubernetes, and there's even FlexVolumes for plug-ins written by vendors but not added to the K8s repositories.

QUESTION 101
A PersistentVolume is a cluster storage resource, but what is a PersistentVolumeClaim? Choose the Answer:

A. A non-durable cluster storage resource
B. A user request for a storage resource
C. A dynamically allocated cluster storage resource
D. A user request for a non-durable storage resource

Correct Answer: B
Explanation

Explanation/Reference:
The PVC request gets matched with a PV.

QUESTION 102
What is ephemeral storage?
Choose the Answer:

A. Storage that can be requested through PVCs
B. Magnetic spinning disk storage on a local host it may not be SSD
C. Local storage on the node used for containers. Its contents are removed when the pod is deleted.
D. Scratch space in a Ceph cluster or other object store

Correct Answer: C
Explanation

Explanation/Reference:
Ephemeral storage is used to run containers on local nodes.

QUESTION 103
What is the lifecycle of PVs and PVCs?
Choose the Answer:

A. Provision, Use, Reclaim
B. Provision, Bind, Use, Reclaim
C. Bind, Provision, Use, Reclaim
D. It depends on whether or not the resource is dynamically provisioned.

Correct Answer: B
Explanation

Explanation/Reference:
First, the administrator provisions the resource or sets it up to be dynamically allocated. Next, when a PVC comes in, it's bound to a PV and used. Once the resource is removed, it's reclaimed according to the reclaim policies.

QUESTION 104
At which levels can you apply security contexts? (Select all that apply) Choose the 2 correct answers:

A. Service
B. Container
C. Pod
D. Deployment

Correct Answer: BC
Explanation

Explanation/Reference:
Containers and pods can have security contexts.

QUESTION 105
What is a Pod Security Policy?
Choose the Answer:

A. An API object representing the pod's allowed container images.
B. A cluster-level resource that controls security sensitive aspects of the pod specification.
C. A way to control which pods can communicate with other pods and the network ports on which communication is permitted.

D. A way to control whether or not the pod can be created by certain roles.

Correct Answer: B
Explanation

Explanation/Reference:
The PodSecurityPolicy objects define a set of conditions that a pod must run with in order to be accepted into the system, as well as defaults for the related fields. They allow an administrator to control many aspects of the pod.

QUESTION 106
Frank is signed in to his Kubernetes cluster and needs a few additional permissions that he does not currently possess to make his application work. How can he accomplish this? Choose the Answer:

A. He can add permissions to a role by simply using kubectl update role
B. He can create a new RBAC request to be approved by an administrator with the command kubectl create rolebinding request
C. He must seek an administrator that has the permissions he needs to get a role created with those permissions.
D. He can create a new role with the permissions he needs and bind that to his user and/or application.

Correct Answer: C
Explanation

Explanation/Reference:
A user cannot escalate his or her own permissions. If a user does not have a given permissions, he or she cannot create a role with those permissions or grant someone else those permissions.

QUESTION 107
Security context settings for a pod include all of the following except Choose the Answer:

A. AppArmor
B. Privileged/Unprivileged
C. SecureImage
D. AllowPrivlegedEscalation

Correct Answer: C
Explanation

Explanation/Reference:
Security contexts for pods allow a lot of specificities but not whether or not the image is secure.

QUESTION 108
How could an administrator limit what a particular user could and could not do in a particular namespace?
Choose the Answer:

A. Configure a new role in the user's namespace with the appropriate rules, then bind the role to the user subject.
B. Configure a new cluster role specifying the rules and applying them to the user object.
C. Configure a network policy with the appropriate permissions and then assign the policy to the user.
D. Configure the user with the correct permission set in the namespace API object.

Correct Answer: A
Explanation

Explanation/Reference:
Roles are API objects, and you don't want to use a ClusterRole for this one since we're specifying a particular namespace. Roles apply to subjects, which can be groups, users, or service accounts.

QUESTION 109
Which of these types of calls should be authenticated and authorized? Choose the Answer:

A. API calls from nodes to the master
B. External service calls from third-party software and devices
C. API calls from users
D. All of these

Correct Answer: D
Explanation

Explanation/Reference:
Everything should be authenticated and authorized. It is not required but should be skipped only with very good reason.

QUESTION 110
What is a way for containers to write information about fatal events to a location, where it can be easily retrieved and surfaced by tools like dashboards and monitoring software, called? Choose the Answer:

A. Node logs
B. Custom logs
C. Termination Messages
D. Annotations

Correct Answer: C
Explanation

Explanation/Reference:
Termination messages, if your containers are configured to write them, can provide valuable information about why a particular pod stopped.

QUESTION 111
When troubleshooting a particular pod, which of these commands is likely to give you the most detailed information?
Choose the Answer:

A. kubectl get pods.

B. kubectl describe pod.
C. kubeadm describe.
D. kubectl get deployment.

Correct Answer: B
Explanation

Explanation/Reference:
A great tool in your kit is kubectl describe. It's very useful and should be one of the first places you look when things head south.

QUESTION 112
You suspect that one of your nodes might be having some difficulty. What is a good command to get an overview of the current status of your cluster? Choose the Answer:

A. kubectl get metrics
B. kubectl get nodes -o wide
C. kubectl cluster-info
D. kubectl top pods

Correct Answer: B
Explanation

Explanation/Reference:
"Get nodes" is a great way to see which of your nodes are in a ready state. Cluster-info doesn't give very much information without the dump flag, and with the flag it gives ALL THE INFORMATION, and would require extensive and tedious review. Kubectl top nodes would also be a valid choice, but top pods wouldn't tell us much of interest about the nodes.

QUESTION 113
How can you run the command /usr/bin/id inside of the pod called "superman"? Choose the Answer:

A. kubectl exec superman /usr/bin/id
B. kubectl run superman /usr/bin/id
C. kubectl exec -it superman /usr/bin/id
D. kubectl exec -it superman /bin/bash to get a shell, then execute the command. There's no way to do it in a single command.

Correct Answer: A
Explanation

Explanation/Reference:
We left out the separator to reinforce that it's optional, but this is the best way.

QUESTION 114
kubectl logs returns what?
Choose the Answer:

A. /var/log/messages from the target container and/or pod.
B. stdout and stderr from the target container and/or pod.
C. Any log specified from the target container and/or pod.
D. /var/log/syslog from the target container and/or pod.

Correct Answer: B
Explanation

Explanation/Reference:
This command will only return stdout and stderr from the containers. Other logs must be accessed differently.

QUESTION 115
LAB1. Deploying Your Cluster

Solution:

First, create a master server using the "Cloudnative Kubernetes" engine. Once this machine has booted, log in to it, change the password, and then start the deployment. K8s requires a pod network to function. We are going to use Flannel, so we need to pass in a flag to the deployment script so K8s knows how to configure itself:

```
sudo kubeadm init pod-network-cidr=10.244.0.0/16
```

This command might take a fair amount of time to complete, possibly as much as ten minutes. Once it's complete, make note of the join command output by kubeadm init that looks like this:

```
kubeadm join token discovery-token-ca-cert-hash sha256:
```

You will run that command on the other nodes to allow them to join the cluster but not quite yet. We'll get to that soon.
Create a directory:

```
mkdir -p $HOME/.kube
```

Next, you'll move the configuration files to a location usable by your local user. if you copy and paste these commands, do so one at a time, or your sudo password prompt might cause things to go slightly wrong and you might have to will be wrong and you might have to redo it.

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

In order for your pods to communicate with one another, you'll need to install pod networking. We are going to use Flannel for our Container Network Interface (CNI) because it's easy to install and reliable. Enter this command:

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.9.1/Documentati on/kube-flannel.yml
```

Next, you'll check to make sure everything is coming up properly.

```
kubectl get pods all-namespaces
```

Once the kube-dns-xxxx containers are up, your cluster is ready to accept worker nodes. Create three or so worker nodes the same way you created your master nodes by bringing up the "Cloudnative Kubernetes" image on your Cloud Servers tab above. ssh to each of the other nodes in the cluster, and execute the kubeadm join command you noted earlier. You will need execute this command with root privileges, so be sure to add "sudo" to the beginning of the command in order for it to complete correctly. Once this command is issued, you may log out of the node. Kubernetes will configure it for you from this point on. See the video "Setting Up Your Cluster" in this course for details and a full walkthrough of the process.
On the master, you can watch the node come up by repeatedly running:

```
kubtctl get nodes
```

LAB2. Run a Job

Applications that run to completion inside a pod are called "jobs." This is useful for doing batch processing.
Most Kubernetes objects are created using yaml. Here is some sample yaml for a job which uses perl to calculate pi to 2000 digits and then stops.

```
apiVersion: batch/v1

kind: Job

metadata:

name: pi

spec:

template:

spec:

containers:

- name: pi

image: perl

command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]

restartPolicy: Never

backoffLimit: 4
```

Create this yaml file on your master node and call it "pi-job.yaml". Run the job with the command:

```
kubectl create -f pi-job.yaml
```

Task:
1. Check the status of the job using the kubectl describe command.
2. When the job is complete, view the results by using the kubectl logs command on the appropriate pod.
3. Write yaml for a new job. Use the image "busybox" and have it sleep for 10 seconds and then complete. Run your job to be sure it works.

Solution:

1. The full command is kubectl describe job pi
2. The previous command will give you the name of the pod associated with the job, which you will need to pass into the kubectl logs command.
For example: (the precise code will vary)

```
$ kubectl describe job pi

Name: pi

Namespace: default

Selector: controller-uid=7ffe0296-f7ad-11e7-8717-0abccbe536d6

Labels: controller-uid=7ffe0296-f7ad-11e7-8717-0abccbe536d6

job-name=pi

Annotations:

Parallelism: 1

Completions: 1

Start Time: Fri, 13 Apr 2018 15:30:20 +0000

Pods Statuses: 0 Running / 1 Succeeded / 0 Failed

Pod Template:

Labels: controller-uid=7ffe0296-f7ad-11e7-8717-0abccbe536d6

job-name=pi

Containers:

pi:

Image: perl

Port:

Command:
perl
```

```
-Mbignum=bpi

-wle

print bpi(2000)

Environment:

Mounts:

Volumes:

Events:

Type Reason Age From Message

-

Normal SuccessfulCreate 4m job-controller Created pod: pi-fmctx

$ kubectl logs pi-fmctx
```

3. The yaml could vary in a couple of ways, but here is an example solution:

```
apiVersion: batch/v1

kind: Job

metadata:

name: busybox

spec:

template:

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "10"]

restartPolicy: Never

backoffLimit: 4
```

LAB3. Deploy a Pod

Pods usually represent running applications in a Kubernetes cluster. Here is an example of some yaml which defines a pod:

```
apiVersion: v1

kind: Pod

metadata:

name: alpine

namespace: default

spec:

containers:

- name: alpine

image: alpine

command:

- sleep

- "3600"

imagePullPolicy: IfNotPresent

restartPolicy: Always
```

Task:
1. Looking at the yaml, describe what the pod will do.
2. Run the pod.
3. Delete the pod.
4. Write yaml for a pod that runs the nginx image.
5. Run your yaml to ensure it functions as expected.
Delete any user pods you created during this lab.

Solution:

1. This pod will cause the alpine linux container to sleep for 3600 seconds (1 hour) and then exit.
Kubernetes will then restart the pod.
2. If the yaml is named alpine.yaml then the command is kubectl create -f alpine.yaml
3. There are a few ways to accomplish this.
Use the file method: kubectl delete -f alpine.yaml

Use the object method: kubectl delete pod alpine or kubectl delete pod/alpine
4. There are many possibilities, but here is yaml that satisfies the exercise:

```
apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx

restartPolicy: Always
```

5. Depending on the file name, the command might be: kubectl create -f nginx-pod.yaml

LAB4. Explore the Sandbox

Task:
1. Examine the current status of your cluster. Are all the nodes ready? How do you know?
2. Are there any pods running on node 2 of your cluster? How can you tell?
3. Is the master node low on memory currently? How can you tell?
4. What pods are running in the kube-system namespace? What command did you use to find out?

Solution:

1. The command kubectl get nodes will give the current status of all nodes.
2. You can get this information in a variety of ways:
kubectl describe node node-name
kubectl get pods all-namespaces -o wide will list all pods and which nodes they are currently running on.
3. kubectl describe node node-2-name will list DiskPressure and MemoryPressure statuses so you can see how it is doing.
4. kubectl get pods -n kube-system will provide the desired results.
LAB5. Deployments

Here is some yaml for an nginx deployment:

```
apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: nginx-deployment

spec:

selector:

matchLabels:

app: nginx

replicas: 2

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.7.9

ports:

- containerPort: 80
```

Task:
1. Create the deployment.
2. Which nodes are the pods running on. How can you tell?
3. Update the deployment to use the 1.8 version of the nginx container and roll it out.
4. Update the deployment to use the 1.9.1 version of the nginxcontainer and roll it out.
5. Roll back the deployment to the 1.8 version of the container.
6. Remove the deployment

Solution:

1. Create the yaml file and name it something. I chose nginx-deployment.yaml. Create the deployment object by calling kubectl create -f nginx-deployment.yaml.
2. You can find this answer by doing a kubectl describe deployment nginx-deployment.
3. There are many ways to get this:
kubectl get pods -l app=nginx -o wide gives you the results in one step and uses a label selector.
Or, you could:
kubectl describe deployment nginx-deployment to get the pod information about the deployment and, using that,

kubectl get pods name-of-pods -o wide
4. There are many ways. Here are two:
kubectl set image deployment nginx-deployment nginx=nginx:1.8. This will work just fine but is not the preferred method because now the yaml is inconsistent with what you've got running in the cluster. Anyone coming across your yaml will assume it's what is up and running and it isn't. Update the line in the yaml to the 1.8 version of the image, and apply the changes with kubectl apply -f nginx-deployment.yaml
5. Same as above. Don't forget you can watch the status of the rollout with the command kubectl rollout status deployment nginx-deployment.
6. kubectl rollout undo deployment nginx-deployment will undo the previous rollout, or if you want to go to a specific point in history, you can view the history with kubectl rollout history deployment nginx-deployment and roll back to a specific state with kubectl rollout history deployment nginx-deployment revision=x.

LAB6. Setting Container Environment Variables

Task:
1. Write yaml for a job that will run the busybox image and will print out its environment variables and shut down.
2. Add the following environment variables to the pod definition:
STUDENT_NAME="Your Name"
SCHOOL="Linux Academy"
KUBERNETES="is awesome"
3. Run the job.
4. Verify that the environment variables were added.

Solution:

1. There are lots of possibilities, but here is what I came up with:

```
apiVersion: v1

kind: Pod
metadata:

name: env-dump

spec:

containers:

- name: busybox

image: busybox

command:

- env
```

2. Change the yaml to something like this:

```
apiVersion: v1

kind: Pod

metadata:

name: env-dump

spec:

containers:

- name: busybox

image: busybox

command:

- env

env:

- name: STUDENT_NAME

value: "Your Name"

- name: SCHOOL

value: "Linux Academy"

- name: KUBERNETES

value: "is awesome"
```

3. kubectl create -f env-dump.yaml and wait for the pod to return the status of "Completed."
4. kubectl logs env-dump will show all the environment variables.
LAB7. Scaling

Consider this YAML for an nginx deployment:

```
apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: nginx-deployment

spec:

selector:
```

```
matchLabels:
app: nginx
replicas: 2
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx
ports:
- containerPort: 80
```

Task:
Complete and answer the following:
1. Scale the deployment up to 4 pods without editing the YAML.
2. Edit the YAML so that 3 pods are available and can apply the changes to the existing deployment.
3.Which of these methods do you think is preferred and why?

Solution:

To scale the deployment up to 4 pods, use: pkubescl scale deployment nginx-deployment replicas=4
To make it so 3 pods can be available and apply the changes to the existing deployment, complete the following:
Edit the YAML as follows:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
name: nginx-deployment
spec:
selector:
matchLabels:
app: nginx
replicas: 3
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx
ports:
- containerPort: 80
```

Execute the command: kubectl -f apply nginx-deployment Performing the edit in the YAML is the preferred one, as it keeps the YAML on disk in sync with the state of the cluster.

LAB8. Replication Controllers, Replica Sets, and Deployments

Deployments replaced the older ReplicationController functionality, but it never hurts to know where you came from. Deployments are easier to work with, and here's a brief exercise to show you how.
A Replication Controller ensures that a specified number of pod replicas are running at any one time. In other words, a Replication Controller makes sure that a pod or a homogeneous set of pods is always up and available.
Write a YAML file that will create a Replication Controller that will maintain three copies of an nginx container. Execute your YAML and make sure it works. A Replica Set is a more advanced version of a Replication Controller that is used when more low-level control is needed. While these are commonly managed with deployments in modern K8s, it's good to have experience with them.
Write the YAML that will maintain three copies of an nginx container using a Replica Set. Test it to be sure it works, then delete it.
A deployment is used to manage Replica Sets.
Write the YAML for a deployment that will maintain three copies of an nginx container. Test it to be sure it works, then delete it.

Solution:

To create the replication controller, write the following YAML file:

Replication Controller:

```
apiVersion: v1
kind: ReplicationController
metadata:
name: nginx
spec:
replicas: 3
selector:
app: nginx
template:
metadata:
name: nginx
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx
ports:
- containerPort: 80
```

To maintain three copies of an nginx container in a replica set, write the following YAML file:
Replication Set:

```
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
name: nginx
labels:
app: nginx
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx
ports:
- containerPort: 80
```

To perform a deployment for the Replica Set, write the following YAML file:
Deployment:

```
apiVersion: apps/v1beta2 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
name: nginx-deployment
labels:
```

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

LAB9. Label ALL THE THINGS!

Putting labels on objects in Kubernetes allow you to identify and select objects in as wide or granular style as you like.
Task:
1. Label each of your nodes with a "color" tag. The master should be black; node 2 should be red; node 3 should be green and node 4 should be blue.
2. If you have pods already running in your cluster in the default namespace, label them with the key/value pair running=beforeLabels.
3. Create a new alpine deployment that sleeps for one minute and is then restarted from a yaml file that you write that labels these container with the key/value pair running=afterLabels.
4. List all running pods in the default namespace that have the key/value pair running=beforeLabels.
5. Label all pods in the default namespace with the key/value pair tier=linuxAcademyCloud.
6. List all pods in the default namespace with the key/value pair running=afterLabels and tier=linuxAcademyCloud.

Solution:

1.
kubectl label node node1-name color=black for the master.
kubectl label node node2-name color=red for node 2.
kubectl label node node3-name color=green for node 3. kubectl label node node4-name color=blue for node 4.
2. kubectl label pods -n default running=beforeLabels all
3. alpine-label.yaml:

apiVersion: v1

kind: Pod

metadata:

name: alpine

namespace: default

labels:

running: afterLabels

spec:

containers:

- name: alpine

image: alpine

command:

- sleep

- "60"

restartPolicy: Always

4. kubectl get pods -l running=beforeLabels -n default
5. kubectl label pods all -n default tier=linuxAcademyCloud
6. kubectl get pods -l running=afterLabels -l tier=linuxAcademyCloud

LAB10. Raise a DaemonSet

No black magic is required, just a bit of yaml.
Write the yaml to deploy a DaemonSet (just use an nginx image) and then test it to be sure it gets deployed on each node. Delete the pods when you've completed this exorcism. Er... Exercise.

Solution:

There are lots of possible solutions to this exercise, but here is what I came up with:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: cthulu
labels:
daemon: "yup"
spec:
selector:
matchLabels:
daemon: "pod"
template:
metadata:
labels:
daemon: pod
spec:
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: cthulu-jr
image: nginx
```

LAB11. Label a Node & Schedule a Pod

Task:
1. Pretend that node 3 is your favorite node. Maybe it's got all SSDs. Maybe it's got a fast network or a GPU. Or maybe it sent you a nice tweet. Label this node in some way so that you can schedule a pod to it.
2. Create a yaml for a busybox sleeper/restarter that will get scheduled to your favorite node from #1.

Solution:

There are many possible answers to this exercise, here is what I came up with:
1. To mark my favorite node, I used kubectl label node node3-name myDarling=bestOne.
2. For my pod to be launched on my favorite node, I used this yaml:

```
apiVersion: v1
kind: Pod
metadata:
name: busybox
namespace: default
spec:
containers:
- name: busybox
image: busybox
command:
- sleep
- "300"
imagePullPolicy: IfNotPresent
restartPolicy: Always
nodeSelector:
myDarling: bestOne
```

LAB12. Multiple Schedulers

Pods generally are scheduled by the default scheduler, and their yaml might look like this:

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

name: annotation-default-scheduler

labels:

name: multischeduler

spec:

schedulerName: default-scheduler

containers:

- name: pod-container

image: k8s.gcr.io/pause:2.0

Ordinarily, we don't need to specify the scheduler's name in the spec because everyone uses a single default one. Sometimes, however, developers need to have custom schedulers in charge of placing pods due to legacy or specialized hardware constraints. Rewrite the yaml for this pod so it makes use of a scheduler called custom-scheduler, and annotate the pod accordingly.

Solution:

apiVersion: v1

kind: Pod

metadata:

name: annotation-default-scheduler

labels:
name: multischeduler

annotations:

scheduledBy: custom-scheduler

spec:

schedulerName: custom-scheduler

containers:

- name: pod-container

image: k8s.gcr.io/pause:2.0

LAB13. View the Logs

Create this object in your cluster:

apiVersion: v1

kind: Pod

metadata:

name: counter

labels:

demo=logger

spec:

containers:

- name: count

image: busybox

args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 3; done']

This is a small container which wakes up every three seconds and logs the date and a count to stdout.
Task:

1. View the current logs of the counter.
2. Allow the container to run for a few minutes while viewing the log interactively.
3. Have the command only print the last 10 lines of the log.
4. Look at the log for the schduler. Have there been any problems lately that you can see?
5. Kubernetes uses etcd for its key-value store. Take a look at its logs and see if it has had any problems lately.
6. Kubernetes API server also runs as a pod in the cluster. Find and examine its logs.

Solution:

1. kubectl logs counter
2. kubectl logs counter -f
3. kubectl logs counter tail=10 or, since tail defaults to 10, just kubectl logs counter tail
4. This question really wants to know if you can find the logs for the scheduler. They're in the master in the /var/log/containers directory. There, a symlink has been create to the appropriate container's log file, and the symlink will have a name that begins with "kube-scheduler-" These logs belong to root, so you will have to sudo to view them.
5. The etcd logs are in the same directory as the logs for the previous question, only the name of the symlink begins with "etcd-", and also belongs to root.
6. The API server also lives in the same directory and begins with "kube-apiserver-", and also belongs to root.

LAB14. Maintenance on Node 3!

Node 3 (Hey, isn't that your favorite node?) needs to have some maintenance done on it. The ionic defibulizer needs a new multiverse battery and the guys in the data center are impatient to get started.
Task:

1. Prepare node 3 for maintenance by preventing the scheduler from putting new pods on to it and evicting any existing pods. Ignore the DaemonSets those pods are only providing services to other local pods and will come back up when the node comes back up.
2. When you think you've done everything correctly, go to the Cloud Servers page and shut node 3 down. Don't delete it! Just stop it. While it's down, we'll pretend that it's getting that new multiverse battery. While you wait for the cluster to stabilize, practice your yaml writing skills by creating yaml for a new deployment that will run six replicas of an image called k8s.gcr.io/pause:2.0. Name this deployment "lots-of-nothing".
3. Bring the "lots-of-nothing" deployment up on your currently 75% active cluster. Notice where the pods land.
4. Imagine you just got a text message from the server maintenance crew saying that the installation is complete. Go back to the Cloud Server tab and start Node 3 up again. Fiddle with your phone and send someone a text message if it helps with the realism. Once Node 3 is back up and running and showing a "Ready" status, allow the scheduler to use it again.
5. Did any of your pods move to take advantage of the additional power? You get 143 bonus points for this exercise if you know what an ionic defibulizer is. Tweet the answer to me @OpenChad. Use the hashtag #NoYouDontReallyGetPoints.

Solution:

1. kubectl drain node3-name ignore-daemonsets
2. Again, there are lots of possible answers, but here's one that I wrote:

```
apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: lots-of-nothing

spec:

selector:

matchLabels:

timeToGet: schwifty

replicas: 6

template:

metadata:

labels:

timeToGet: schwifty

spec:

containers:

- name: pickle-rick

image: k8s.gcr.io/pause:2.0
```

3. kubectl create -f lots-of-nothing.yaml will bring it up if you named the yaml the same way I did. If you set up your labels like I did, then the command to show you where all the pods wound up is kubectl get pods -o wide -l timeToGet=schwifty. My point here is beyond just being a little silly your labels can be whatever you want them to, so it makes a lot of sense to think through some conventions and standards with your colleagues and other users of your cluster, otherwise you will end up with nonsensical labels like mine.
4.kubectl uncordon node3-name will allow the scheduler to once again allow pods to be scheduled on the node.
5. No, the uncordon only affects pods being scheduled and won't move any back unless other nodes are experiencing MemoryPressure or DiskPressure.

LAB15. Cluster DNS & Service Discovery

In a Kubernetes cluster, services discover one another through the Cluster DNS. Names of services resolve to their ClusterIP, allowing application developers to only know the name of the service deployed in the cluster and not have to figure out how to get all the right IP addresses into the right containers at deploy time.
Here is yaml for a deployment:

```
apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: bit-of-nothing

spec:

selector:

matchLabels:

app: pause

replicas: 2

template:

metadata:

labels:

app: pause

spec:
```

```
containers:

- name: bitty

image: k8s.gcr.io/pause:2.0
```

Create this file and name it bit-of-nothing.yaml.
Task:

1. Run the deployment and verify that the pods have started.
2. Start a busybox pod you can use to check DNS resolution (you can use my yaml, below, if you like, but it's good practice to write your own!)

```
apiVersion: v1

kind: Pod

metadata:

name: busybox

namespace: default

spec:

containers:

- name: busybox

image: busybox

command:

- sleep

- "3600"

imagePullPolicy: IfNotPresent

restartPolicy: Always
```

3. Check to see whether or not bit-of-nothing is currently being resolved in the cluster via your busybox container.
4. Expose the bit-of-nothing deployment as a ClusterIP service.
5. Verify that "bit-of-nothing" is now being resolved to an IP address in your cluster.

Solution:

1. kubectl create -f bit-of-nothing.yaml At this point, this command should be second nature to you, unless you're doing a lot of skipping around.
2. kubectl create -f busybox.yaml
3.kubectl exec -it busybox nslookup bit-of-nothing should error out with not found. 4.kubectl expose deployment bit-of-nothing type=ClusterIP port 80
5. kubectl exec -it busybox nslookup bit-of-nothing should return an IP address after a few moments.
LAB16. Work with Persistent Storage

Task:
1. On one of your cloud servers, install and configure Ubuntu 16 to be an NFS server. Export a directory for use by the cluster. Be sure you give all of your cluster nodes access to the directory.
2. Configure each of your Kubernetes nodes including the master as NFS clients.
3. Write the yaml to provision the storage in Kubernetes and call the API with it, and verify that the appropriate object has been created.
4. Write the yaml to request usage of the storage and call the API. Verify that the appropriate objects were created and/or allocated.
5. Write the yaml required by a busybox container to mount the volume at /tmp. Call the api.
6. Invoke the shell on the busybox to interactively work with the container. Change to the /tmp directory and attempt to create a file called "its-alive.txt".
7. Verify that the file exists on the NFS server in the exported directory.

Solution:

1. Once the NFS node comes up, perform the following steps on it:
sudo apt update
sudo apt upgrade -y
Make a note of the internal IP addresses your four Kubernetes cluster are assigned.
sudo apt install nfs-kernel-server -y
Create a directory: sudo mkdir /var/nfs/general -p
Set correct owner and group on the directory: sudo chown nobody:nogroup /var/nfs/general Edit the file /etc/exports such that the following line is added:

/var/nfs/general x.x.x.x(rw,sync,no_subtree_check) y.y.y.y(rw,sync,no_subtree_check) z.z.z.z(rw,s ync,no_subtree_check) a.a.a.a(rw,sync,no_subtree_check)

Where x.x.x.x, y.y.y.y, z.z.z.z, and a.a.a.a are the internal IP addresses of your Kubernetes nodes.
Restart the NFS server with the command sudo systemctl restart nfs-kernel-server.
2. Log in to each of your Kubernetes nodes, including the master, and execute the command sudo apt install nfs-common
3. Make a note of your NFS server's internal ip address and create this file,pv.yaml, while replacing b.b.b.b with the NFS server's IP address:

```
apiVersion: v1

kind: PersistentVolume

metadata:

name: lab-vol
spec:

capacity:

storage: 1Gi

volumeMode: Filesystem

accessModes:
```

- ReadWriteMany

persistentVolumeReclaimPolicy: Recycle

nfs:

path: /var/nfs/general

server: b.b.b.b

readOnly: false

Execute the command kubectl create -f pv.yaml and verify that the PersistentVolume was provisioned by executing kubectl get pv
4. Create the file for the PVC similar to the following yaml file named pvc.yaml:

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nfs-pvc

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

Execute the command kubectl create -f pvc.yaml and verify that it was created and bound to the lab-vol PV with the command kubectl get pvc and another look at kubectl get pv should also verify that they two objects are bound.
5. I created this yaml called nfs-pod.yaml:

apiVersion: v1

kind: Pod
metadata:

name: nfs-pod

labels:

name: nfs-pod

spec:

containers:

- name: nfs-ctn

image: busybox

command:

- sleep

- "3600"

volumeMounts:

- name: nfsvol

mountPath: /tmp

restartPolicy: Always

securityContext:

fsGroup: 65534

runAsUser: 65534

volumes:

- name: nfsvol

persistentVolumeClaim:

claimName: nfs-pvc

And created the pod by executing kubectl create -f nfs-pod.yaml
6. Invoke the shell in the busybox container by using the command kubectl exec -it nfs-pod sh and then running these commands:
cd /tmp
echo "My Monster!" > its-alive.txt
exit
7. On the NFS server, verify that the file was written:ls -la /var/nfs/general/ LAB17.

Task:
1. Create a job that calculates pi to 2000 decimal points using the container with the image named perl and the following entry point to the container:

["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]

Once the job has completed, save the result to your home directory in a file called pi.result.txt.

2. Create yaml for a pod that uses the nginx image and listens on port 80. The pod should have the name nginx-pod and be labeled with app: nginx. Save the yaml to your home directory as nginx-pod.yaml and start the pod.

3. Create yaml for a deployment of two replicas of nginx, listening on the container's port 80. They should bear the label of tier=frontend and app-nginx. The deployment should be named nginx-deployment. Leave a copy of the yaml in your home directory with the name nginx-deployment.yaml.

4. Create a pod called "with-files" with an nginx image listening on port 80. The pod should attach to emptyDir storage, mounted to /tmp in the container. Connect to the pod and create a file with zero bytes in the /tmp directory called "linuxacademy.txt." Do not delete this pod. If you create other artifacts in the course working on this, you may delete them from your home directory or create a directory called "extras" in your home directory and move the files there.

5. Label the worker node of your cluster with rack=qa.

6. Create a file called counter.yaml in your home directory and paste the following yaml into it:

apiVersion: v1

kind: Pod

metadata:

name: counter

spec:

containers:

- name: count

image: busybox

args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']

Start this pod. Once its logs exceed a count of 20 (no need to be precise, any time after it has reached 20 is fine) save the logs into a file in your home directory called count.result.txt. Delete the pod.

7. Create a deployment with two replicas of nginx:1.7.9. The container listens on port 80. It should be named "web-dep" and be labeled with tier=frontend with an annotation of AppVersion=3.4. The containers must be running with the UID of 1000.

8. Upgrade the image in use by the web-dep deployment to nginx:1.9

9. Roll the image in use by the web-dep deployment to the previous version. Do not set the version number of the image explicitly for this command.

10. Scale the number of replicas of the web-dep deployment up to 3.

11. Expose the web-dep deployment as a service using a NodePort.

12. Configure a DaemonSet to run the image k8s.gcr.io/pause:2.0 in the cluster.

13. Configure the cluster to use 8.8.8.8 and 8.8.4.4 as upstream DNS servers.

14. A legacy application (yaml below) requires that the IP address or host name of the web-dep endpoint be configured as an environment variable called "FROBOZZ" in the container. Alter the yaml accordingly, leave a copy in your home directory with the name zork.yaml, and start the pod.

apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: zork

namespace: default

spec:

selector:

matchLabels:

app: zork-app

template:

metadata:

labels:

app:zork-app
spec:

containers:

- name: p114

image: k8s.gcr.io/pause:2.0

15. Create a pod using the k8s.gcr.io/pause:20 container image that is allowed to run on the master node if necessary, but does not have to be scheduled there.

16. Copy all Kubernetes Scheduler logs into a logs directory in your home directory.

17. Run the pod below until the counter in its stdout exceeds 20 (no need to be precise) and then extract the legacy log file to the logs directory in your home directory. Delete the pod.

```
apiVersion: v1
kind: Pod
metadata:
name: counter2
spec:
containers:
- name: count
image: busybox
args: [/bin/sh, -c, 'i=0; while true; do echo "$i: $(date)"; echo "$(date) - File - $i" >> /var/www /countlog; i=$((i+1)); sleep 3; done']
```

18. Build a default network policy that disallows all traffic to pods in the default namespace.

19. Remove the taint from the master node so it can now accept any work even from nodes with no tolerations of the master node.

20. Create a yaml file for a secret called my-secret and saved in a file in your home directory called my-secret.yaml. The secret should have two fields: a username and password. The username should be set to "admin" and the password should be set to "iHeartKittens"

Solution:

1. pi.yaml to create the job is:

```
apiVersion: batch/v1
kind: Job
metadata:
name: pi
spec:
template:
spec:
containers:
- name: pi
image: perl
command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
restartPolicy: Never
backoffLimit: 4
```

After it completes, you'll execute a kubectl logs pi > pi.result.txt

2. nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
name: nginx-pod
labels:
app: nginx
spec:
containers:
- name:
image: nginx
ports:
containerPort: 80
```

3. nginx-deployment.yaml:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
name: nginx-deployment
spec:
selector:
matchLabels:
```

```
app: nginx
replicas: 2
template:
metadata:
labels:
app: nginx
tier: frontend
spec:
containers:
- name: nginx
image: nginx:1.7.9
ports:
- containerPort: 80
```

4. with-files.yaml:

```
apiVersion: v1
kind: Pod
metadata:
name: with-files
spec:
containers:
- image: nginx
name: write-files
volumeMounts:
- mountPath: /tmp
name: temp-volume
volumes:
- name: temp-volume
emptyDir: {}
```

5. kubectl label node node-name rack=qa
6. Use kubectl logs counter > count.result.txt
7. web-dep.yaml:

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
name: web-dep
annotations:
AppVersion: "3.4"
spec:
selector:
matchLabels:
app: nginx
replicas: 2
template:
metadata:
labels:
app: nginx
tier: frontend
spec:
runAsUser: 1000
```

```
containers:
- name: nginx

image: nginx:1.7.9

ports:

- containerPort: 80
```

8. kubectl set image deployment web-dep nginx=1.9
9. kubectl rollout undo deployment web-dep
10.kubectl scale deployment web-dep replicas=3
11. kubectl expose deployment web-dep type=NodePort containerPort 80
12. ds.yaml

```
apiVersion: apps/v1

kind: DaemonSet

metadata:

name: zuul

spec:

selector:

matchLabels:

quiet: "pod"

template:

metadata:

labels:

quiet: pod

spec:

tolerations:

- key: node-role.kubernetes.io/master

effect: NoSchedule

containers:

- name: gozer

image: k8s.gcr.io/pause:2.0
```

13.kube-dns.yaml:

```
apiVersion: v1
kind: ConfigMap

metadata:

name: kube-dns

namespace: kube-system

data:

upstreamNameservers: |

["8.8.8.8", "8.8.4.4"]
```

14. zork.yaml:

```
apiVersion: apps/v1beta2

kind: Deployment

metadata:

name: zork

namespace: default

spec:

selector:

matchLabels:

app: zork-app

template:

metadata:

labels:
```

```
app:zork-app
spec:
containers:
- name: p114
image: k8s.gcr.io/pause:2.0
env:
- name: FROBOZZ
value: "web-dep.default"
```

15. spy.yaml:

```
apiVersion: v1
kind: Pod
metadata:
name: spy
spec:
tolerations:
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: sneak
image: k8s.gcr.io/pause:20
```

16. The Kubernetes logs are located in /var/log/pods/
17. kubtctl exec -it counter2 cat /var/www/countlog > ~/home/logs/countlog
18. net-policy.yaml

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: default-deny
spec:
podSelector: {}
policyTypes:
- Ingress
```

19.kubectl taint node master-node-name node-role.kubernetes.io/master-
20. my-secret.yaml:

```
apiVersion: v1
kind: Secret
metadata:
name: my-secret
type: Opaque
data:
username: YWRtaW4=
password: aUhlYXJ0S2l0dGVucw==
```

Don't forget to run your secrets through base64!

A.

Correct Answer:
Explanation

Explanation/Reference: