LinuxFoundation.CKA.v2026-06-18.q88

Exam Code:	CKA
Exam Name:	Certified Kubernetes Administrator (CKA) Program Exam
Certification Provider:	Linux Foundation
Free Question Number:	88
Version:	v2026-06-18
# of views:	110
# of Questions views:	880
https://www.exam-tests.com/CKA-exam/LinuxFoundation.CKA.v2026-06-18.q88.html	

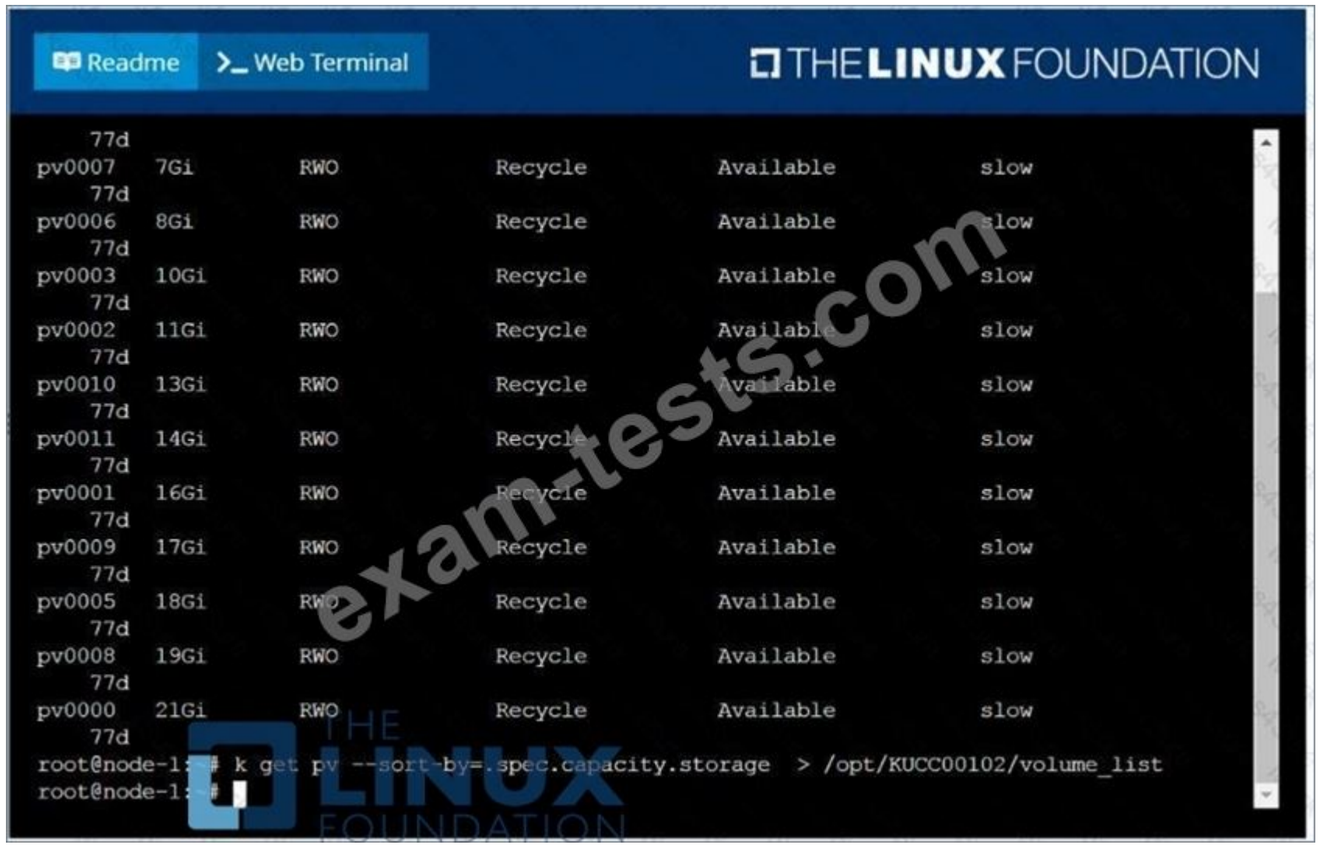
NEW QUESTION: 1
List "nginx-dev" and "nginx-prod" pod and delete those pods
Answer:
See the solution below.
Explanation
kubect1 get pods -o wide
kubectl delete po "nginx-dev"kubectl delete po "nginx-prod"

NEW QUESTION: 2
Score: 4%

Task
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 .
Answer:
See the solution below.
Explanation
Solution:
kubectl run kucc8 --image=nginx --dry-run -o yaml > kucc8.yaml
vi kucc8.yaml
apiVersion: v1
kind: Pod
metadata:

creationTimestamp: null
name: kucc8
spec:
containers:
- image: nginx
name: nginx
- image: redis
name: redis
- image: memcached
name: memcached
- image: consul
name: consul
#
kubectl create -f kucc8.yaml
#12.07

NEW QUESTION: 3
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.
Answer:



NEW QUESTION: 4

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

Answer:

Persistent Volume

A persistent volume is a piece of storage in a Kubernetes cluster. PersistentVolumes are a cluster-level resource like nodes, which don't belong to any namespace. It is provisioned by the administrator and has a particular file size. This way, a developer deploying their app on Kubernetes need not know the underlying infrastructure. When the developer needs a certain amount of persistent storage for their application, the system administrator configures the cluster so that they consume the PersistentVolume provisioned in an easy way.

Creating Persistent Volume

kind: PersistentVolume

apiVersion: v1

metadata:

name:app-data

spec:

capacity: # defines the capacity of PV we are creating

storage: 2Gi #the amount of storage we are tying to claim

accessModes: # defines the rights of the volume we are creating

- ReadWriteMany

hostPath:
path: "/srv/app-data" # path to which we are creating the volume
Challenge
* Create a Persistent Volume named app-data, with access mode ReadWriteMany, storage classname shared, 2Gi of storage capacity and the host path /srv/app-data.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
      path: /srv/app-data
  storageClassName: shared
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"app-data.yaml" 12L, 194C
```


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

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

3. View the persistent volume.

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl get pv
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
app-data   2Gi        RWX            Retain           Available           shared                 31s
```

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

PersistentVolumeClaim
In a real ecosystem, a system admin will create the PersistentVolume then a developer will create a PersistentVolumeClaim which will be referenced in a pod. A PersistentVolumeClaim is created by specifying the minimum size and the access mode they require from the persistentVolume.
Challenge
* Create a Persistent Volume Claim that requests the Persistent Volume we had created above. The claim should request 2Gi. Ensure that the Persistent Volume Claim has the same storageClassName as the persistentVolume you had previously created.
kind: PersistentVolume
apiVersion: v1
metadata:
name:app-data
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 2Gi
storageClassName: shared
2. Save and create the pvc
njerry191@cloudshell:~ (extreme-clone-2654111)$ kubect1 create -f app-data.yaml persistentvolumeclaim/app-data created
3. View the pvc
Image for post

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

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

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

Our status has now changed from available to bound.
5. Create a new pod named myapp with image nginx that will be used to Mount the Persistent Volume Claim with the path /var/app/config.
Mounting a Claim
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
name: app-data
spec:

volumes:
- name:congigpvc
persistenVolumeClaim:
claimName: app-data
containers:
- image: nginx
name: app
volumeMounts:
- mountPath: "/srv/app-data "
name: configpvc

NEW QUESTION: 5
Scale the deployment webserver to
Answer:
See the solution below.
Explanation
solution

NEW QUESTION: 6

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.

Answer:

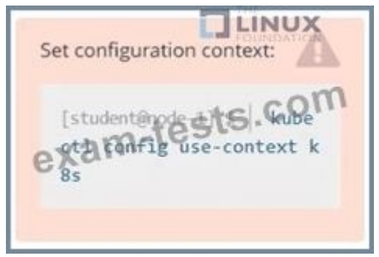
See the solution below.

Explanation

solution

NEW QUESTION: 7

Score: 4%



Task

Check to see how many nodes are ready schedulable (not including nodes tainted NoSchedule) and write the number to /opt/KUSC00402/kusc00402.txt.

Answer:
Solution:
kubectl describe nodes | grep ready|wc -l
kubectl describe nodes | grep -i taint | grep -i noschedule |wc -l
echo 3 > /opt/KUSC00402/kusc00402.txt
#
kubectl get node | grep -i ready |wc -l
taints#noSchedule
kubectl describe nodes | grep -i taints | grep -i noschedule |wc -l
#
echo 2 > /opt/KUSC00402/kusc00402.txt

NEW QUESTION: 8
Create a persistent volume with name app-data, of capacity 2Gi and access mode ReadWriteMany. The type of volume is hostPath and its location is /srv/app-data.
Answer:
solution
Persistent Volume
A persistent volume is a piece of storage in a Kubernetes cluster. PersistentVolumes are a cluster-level resource like nodes, which don't belong to any namespace. It is provisioned by the administrator and has a particular file size. This way, a developer deploying their app on Kubernetes need not know the underlying infrastructure. When the developer needs a certain amount of persistent storage for their application, the system administrator configures the cluster so that they consume the PersistentVolume provisioned in an easy way.
Creating Persistent Volume
kind: PersistentVolume apiVersion: v1 metadata: name:app-data spec: capacity: # defines the capacity of PV we are creating storage: 2Gi #the amount of storage we are tying to claim accessModes: # defines the rights of the volume we are creating - ReadWriteMany hostPath: path: "/srv/app-data" # path to which we are creating the volume Challenge Create a Persistent Volume named app-data, with access mode ReadWriteMany, storage classname shared, 2Gi of storage capacity and the host path /srv/app-data.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: app-data
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
      path: /srv/app-data
  storageClassName: shared
~
~
~
~
~
~
~
~
~
~
~
~
"app-data.yaml" 12L, 194C
```


2. Save the file and create the persistent volume.

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

3. View the persistent volume.

```
njerry191@cloudshell:~ (extreme-clone-265411)$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
app-data  2Gi        RWX            Retain           Available           shared                  31s
```

Our persistent volume status is available meaning it is available and it has not been mounted yet. This status will change when we mount the persistentVolume to a persistentVolumeClaim.
PersistentVolumeClaim
In a real ecosystem, a system admin will create the PersistentVolume then a developer will create a PersistentVolumeClaim which will be referenced in a pod. A PersistentVolumeClaim is created by specifying the minimum size and the access mode they require from the persistentVolume.
Challenge
Create a Persistent Volume Claim that requests the Persistent Volume we had created above. The claim should request 2Gi. Ensure that the Persistent Volume Claim has the same storageClassName as the persistentVolume you had previously created.
kind: PersistentVolume apiVersion: v1 metadata: name:app-data
spec:
accessModes: - ReadWriteMany resources:
requests: storage: 2Gi
storageClassName: shared
2. Save and create the pvc
njerry191@cloudshell:~ (extreme-clone-2654111)$ kubect1 create -f app-data.yaml persistentvolumeclaim/app-data created
3. View the pvc

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

4. Let's see what has changed in the pv we had initially created.

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

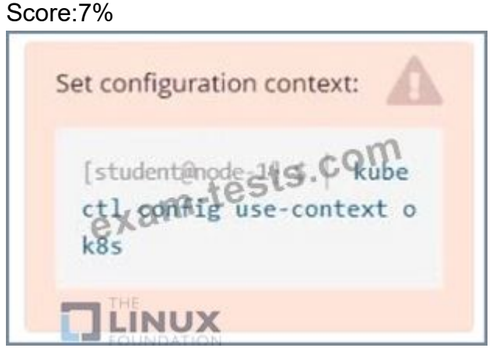
Our status has now changed from available to bound.
5. Create a new pod named myapp with image nginx that will be used to Mount the Persistent Volume Claim with the path /var/app/config.
Mounting a Claim
apiVersion: v1 kind: Pod metadata: creationTimestamp: null name: app-data spec: volumes: - name:congigpvc persistenVolumeClaim: claimName: app-data containers: - image: nginx name: app volumeMounts: - mountPath: "/srv/app-data " name: configpvc

NEW QUESTION: 9
You must connect to the correct host.
Failure to do so may result in a zero score.
[candidate@base] $ ssh Cka000049
Task
Perform the following tasks:
Create a new PriorityClass named high-priority for user-workloads with a value that is one less than the highest existing user-defined priority class value.
Patch the existing Deployment busybox-logger running in the priority namespace to use the high-priority priority class.
Answer:
Task Summary

* SSH into the correct node: cka000049
* Find the highest existing user-defined PriorityClass
* Create a new PriorityClass high-priority with a value one less
* Patch Deployment busybox-logger (in namespace priority) to use this new PriorityClass Step-by-Step Solution
1## SSH into the correct node
bash
CopyEdit
ssh cka000049
Skipping this = zero score
2## Find the highest existing user-defined PriorityClass
Run:
bash
CopyEdit
kubectl get priorityclasses.scheduling.k8s.io
Example output:
vbnet
CopyEdit
NAME VALUE GLOBALDEFAULT AGE
default-low 1000 false 10d
mid-tier 2000 false 7d
critical-pods 1000000 true 30d
Exclude system-defined classes like system-* and the default global one (e.g., critical-pods).
Let's assume the highest user-defined value is 2000.
So your new class should be:
* Value = 1999
3## Create the high-priority PriorityClass
Create a file called high-priority.yaml:
cat <<EOF > high-priority.yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
name: high-priority
value: 1999
globalDefault: false
description: "High priority class for user workloads"
EOF
Apply it:
kubectl apply -f high-priority.yaml
4## Patch the busybox-logger deployment
Now patch the existing Deployment in the priority namespace:
kubectl patch deployment busybox-logger -n priority \

--type='merge' \
-p '{"spec": {"template": {"spec": {"priorityClassName": "high-priority"}}}}'
5## Verify your work
Confirm the patch was applied:
kubectl get deployment busybox-logger -n priority -o jsonpath='{.spec.template.spec.priorityClassName}'
You should see:
high-priority
Also, confirm the class exists:
kubectl get priorityclass high-priority
Final Command Summary
ssh cka000049
kubectl get priorityclass
Create the new PriorityClass
cat <<EOF > high-priority.yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
name: high-priority
value: 1999
globalDefault: false
description: "High priority class for user workloads"
EOF
kubectl apply -f high-priority.yaml
Patch the deployment
kubectl patch deployment busybox-logger -n priority \
--type='merge' \
-p '{"spec": {"template": {"spec": {"priorityClassName": "high-priority"}}}}'
Verify
kubectl get deployment busybox-logger -n priority -o jsonpath='{.spec.template.spec.priorityClassName}' kubectl get priorityclass high-priority

NEW QUESTION: 10
Score:7%



Task
Create a new PersistentVolumeClaim
* Name: pv-volume
* Class: csi-hostpath-sc
* Capacity: 10Mi
Create a new Pod which mounts the PersistentVolumeClaim as a volume:
* Name: web-server
* Image: nginx
* Mount path: /usr/share/nginx/html
Configure the new Pod to have ReadWriteOnce access on the volume.
Finally, using kubectl edit or kubectl patch expand the PersistentVolumeClaim to a capacity of 70Mi and record that change.
Answer:
Solution:
vi pvc.yaml
storageclass pvc
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: pv-volume
spec:
accessModes:
- ReadWriteOnce
volumeMode: Filesystem
resources:
requests:
storage: 10Mi
storageClassName: csi-hostpath-sc
vi pod-pvc.yaml
apiVersion: v1
kind: Pod
metadata:
name: web-server
spec:
containers:
- name: web-server
image: nginx
volumeMounts:
- mountPath: "/usr/share/nginx/html"
name: my-volume
volumes:
- name: my-volume

persistentVolumeClaim:
claimName: pv-volume
craete
kubectl create -f pod-pvc.yaml
#edit
kubectl edit pvc pv-volume --record

NEW QUESTION: 11
Check the image version in pod without the describe command
Answer:
kubectl get po nginx -o
jsonpath='{.spec.containers[].image}{"\n"}'

NEW QUESTION: 12
List all the pods showing name and namespace with a json path expression
Answer:
kubectl get pods -o=jsonpath="{.items[*]['metadata.name',
'metadata.namespace']}"

NEW QUESTION: 13
Create a Pod nginx and specify both CPU, memory requests and limits together and verify.
A. kubectl run nginx-request --image=nginx --restart=Always --dryrun -o yaml > nginx-request.yml
// add the resources section and create
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx-request
spec:
containers:
- image: nginx
name: nginx
resources:
requests:
memory: "100Mi"
cpu: "0.5"
limits:
memory: "200Mi"
cpu: "1"
restartPolicy: Always

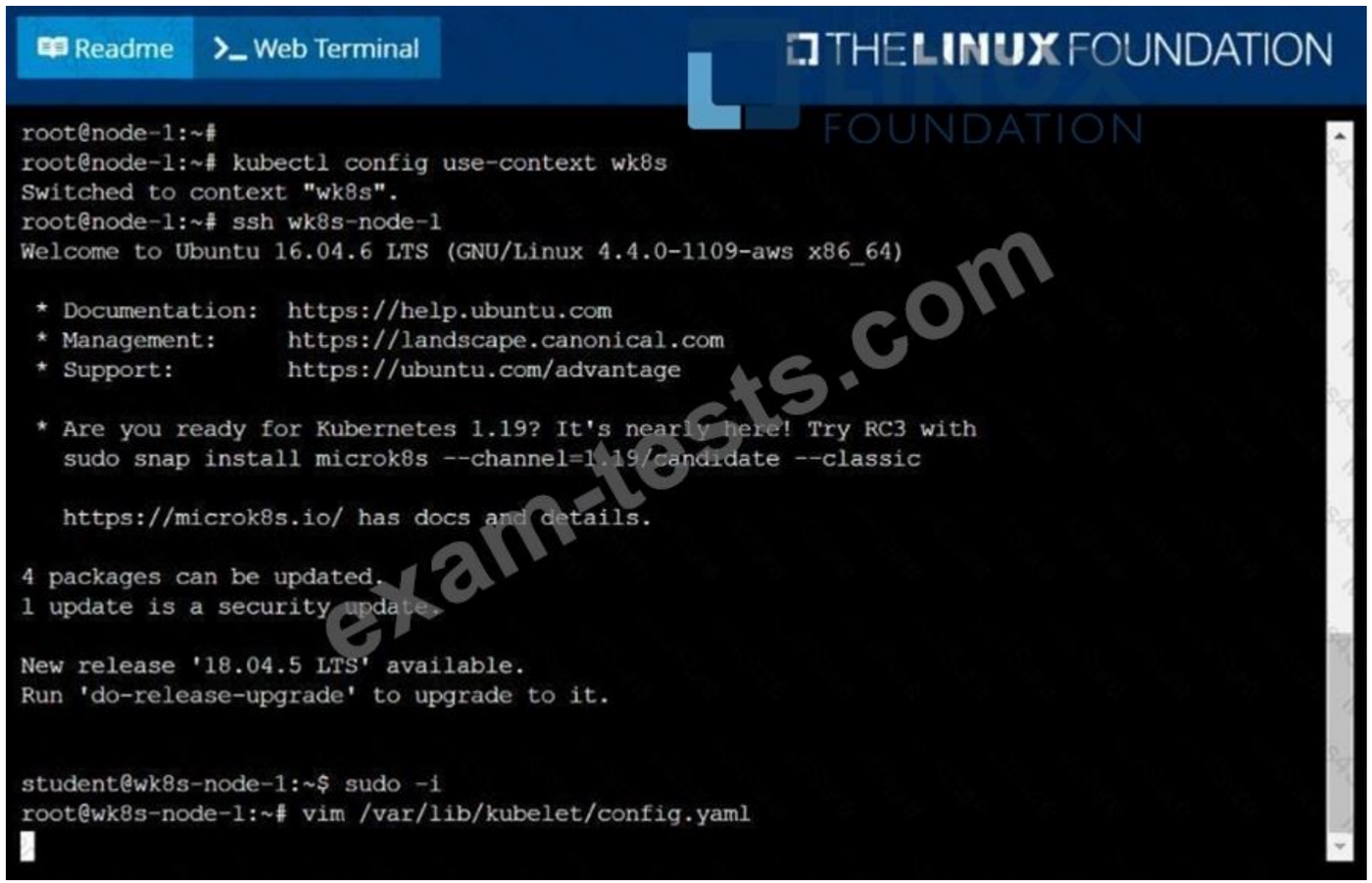
k kubectl apply -f nginx-request.yaml
// Verify
Kubectl top po
B. kubectl run nginx-request --image=nginx --restart=Always --dryrun -o yaml > nginx-request.yml
// add the resources section and create
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
resources:
requests:
memory: "100Mi"
cpu: "0.4"
limits:
memory: "200Mi"
cpu: "7"
restartPolicy: Always
k kubectl apply -f nginx-request.yaml
// Verify
Kubectl top po
Answer: A (LEAVE A REPLY)

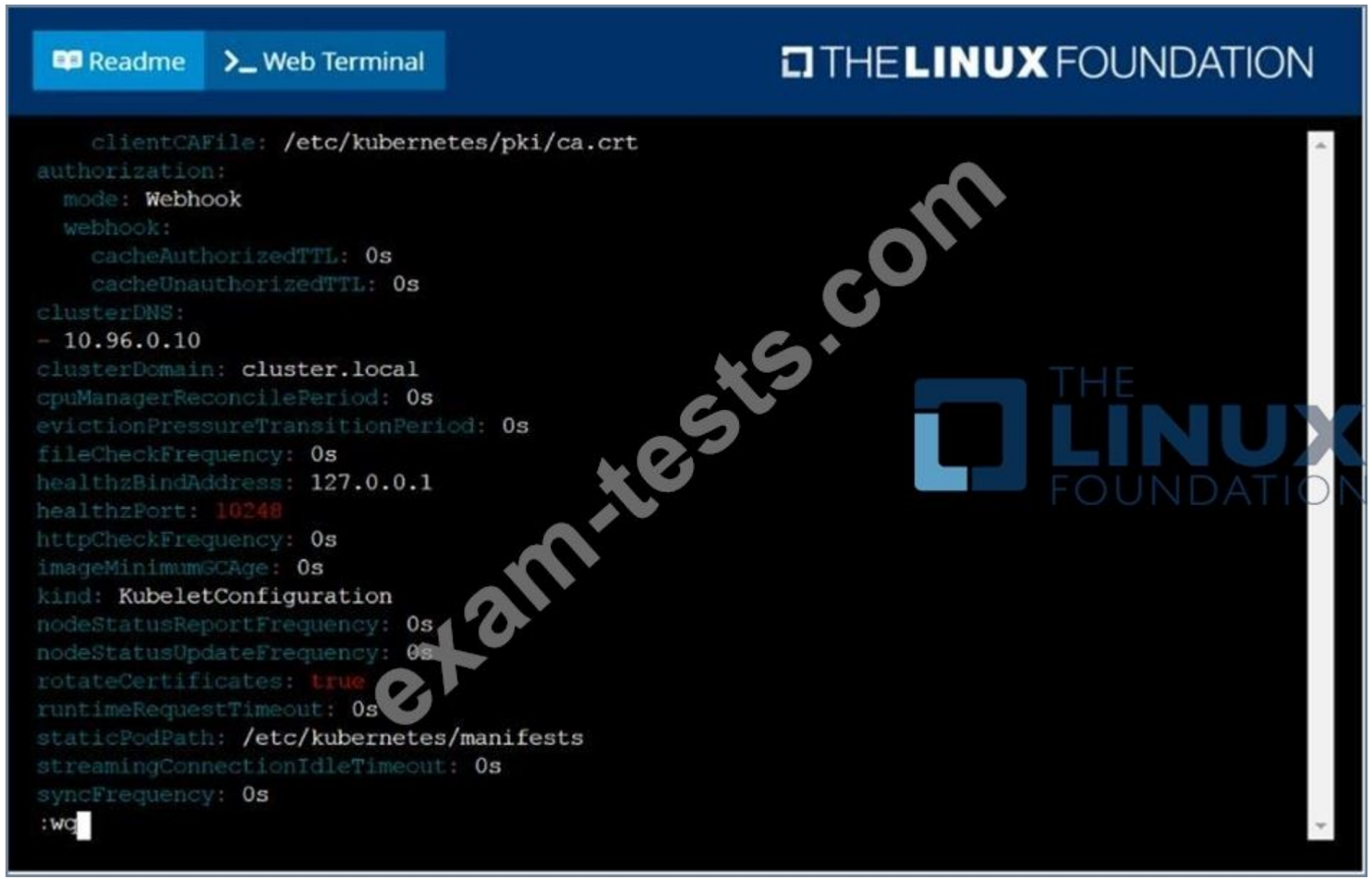
NEW QUESTION: 14
Check the history of deployment
Answer:
kubectl rollout history deployment webapp

NEW QUESTION: 15
For this item, you will have to ssh and complete all tasks on these
nodes. Ensure that you return to the base node (hostname:) when you have completed this item.
Context
As an administrator of a small development team, you have been asked to set up a Kubernetes cluster to test the viability of a new application.
Task
You must use kubeadm to perform this task. Any kubeadm invocations will require the use of the
--ignore-preflight-errors=all option.
Configure the node ik8s-master-O as a master node. .
Join the node ik8s-node-o to the cluster.
Answer:
See the solution below.

Explanation
solution
You must use the kubeadm configuration file located at /etc/kubeadm.conf when initializingyour cluster.
You may use any CNI plugin to complete this task, but if you don't have your favourite CNI plugin's manifest URL at hand, Calico is one popular option:
https://docs.projectcalico.org/v3.14/manifests/calico.yaml
Docker is already installed on both nodes and apt has been configured so that you can install the required tools.

NEW QUESTION: 16
Configure the kubelet systemd- managed service, on the node labelled with name=wk8s-node-1, to launch a pod containing a single container of Image httpd named webtool automatically. Any spec files required should be placed in the /etc/kubernetes/manifests directory on the node.
You can ssh to the appropriate node using:
[student@node-1] $ ssh wk8s-node-1
You can assume elevated privileges on the node with the following command:
[student@wk8s-node-1] $ | sudo -i
Answer:




Readme
Web Terminal
THE LINUX FOUNDATION
clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
mode: Webhook
webhook:
cacheAuthorizedTTL: 0s
cacheUnauthorizedTTL: 0s
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
:wq
exam-tests.com
THE LINUX FOUNDATION

Readme | Web Terminal

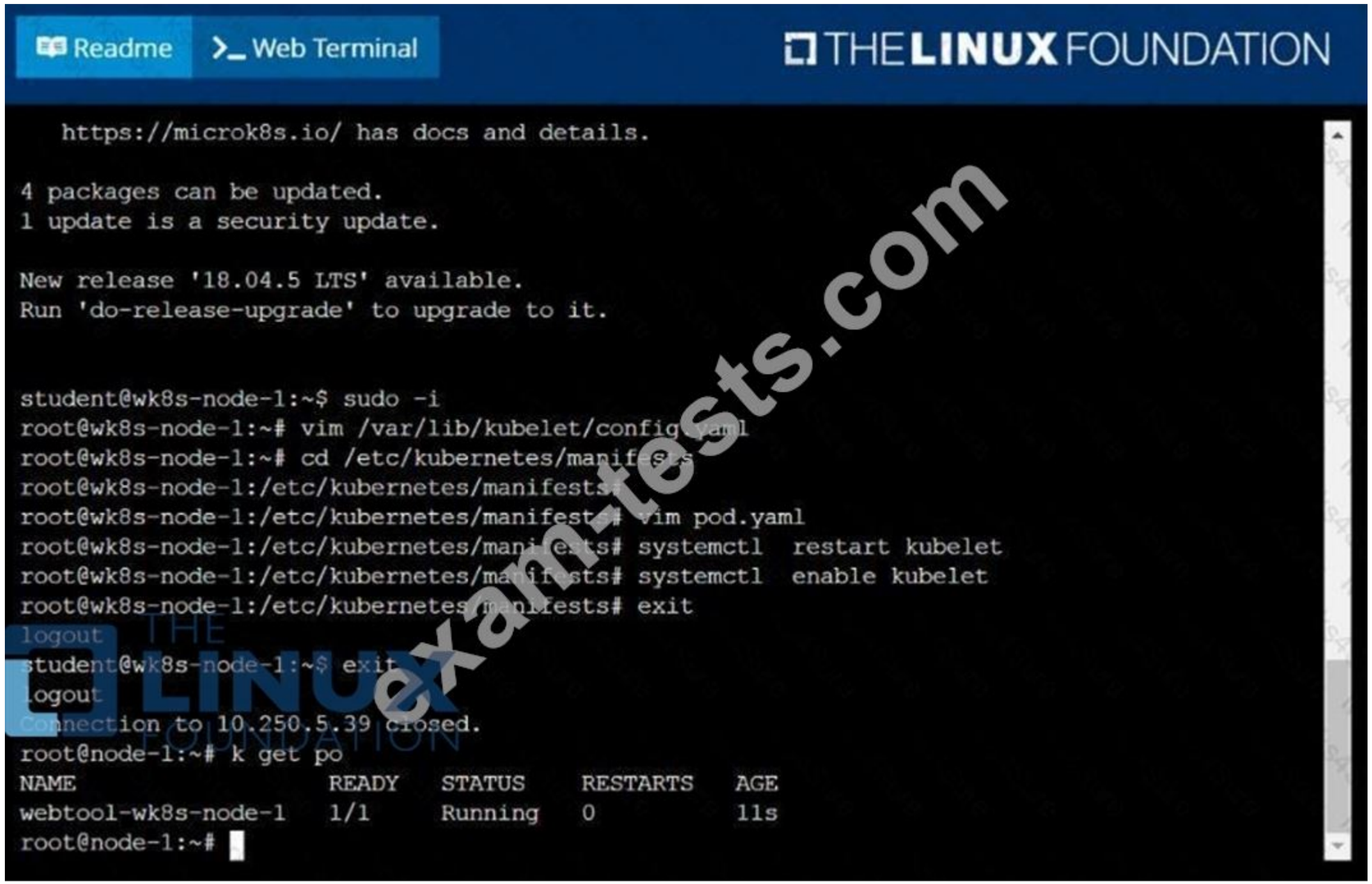
THE LINUX FOUNDATION

```
root@node-1:~# ssh wk8s-node-1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

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

  Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
  sudo snap install microk8s --channel=1.19/candidate --classic

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

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

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


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

Readme | Web Terminal

THE LINUX FOUNDATION

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



NEW QUESTION: 17
List all service account and create a service account called "admin"
A. kubectl get sa
kubectl get sa --all-namespaces
kubectl create sa admin

//Verify
kubectl get sa admin -o yaml
B. kubectl get sa
kubectl get sa --all-namespaces
//Verify
kubectl get sa admin -o yaml
Answer: A (LEAVE A REPLY)

NEW QUESTION: 18
List "nginx-dev" and "nginx-prod" pod and delete those pods
Answer:
kubect1 get pods -o wide
kubectl delete po "nginx-dev"
kubectl delete po "nginx-prod"

NEW QUESTION: 19
Score: 5%

Task
Monitor the logs of pod bar and:
* Extract log lines corresponding to error file-not-found
* Write them to /opt/KUTR00101/bar
Answer:
Solution:
kubectl logs bar | grep 'unable-to-access-website' > /opt/KUTR00101/bar cat /opt/KUTR00101/bar

NEW QUESTION: 20
Create a redis pod and expose it on port 6379
A. kubectl run redis --image=redis --restart=Never --port=6379
YAML File :
apiVersion: v1
kind: Pod
metadata:
labels:
run: redis

name: redis
spec:
containers:
ports:
- containerPort: 6679
Rt restartPolicy: Alwaysf
B. kubectl run redis --image=redis --restart=Never --port=6379
YAML File :
apiVersion: v1
kind: Pod
metadata:
labels:
run: redis
name: redis
spec:
containers:
- image: redis
name: redis
ports:
- containerPort: 6379
Rt restartPolicy: Always
Answer: B (LEAVE A REPLY)

NEW QUESTION: 21
Create a nginx pod with label env=test in engineering namespace
See the solution below.
A. kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml > nginx-pod.yaml kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml | kubectl create -n engineering -f - YAML File:
apiVersion: v1
kind: Pod
metadata:
name: nginx
namespace: engineering
labels:
env: test
spec:
containers:
- name: nginx
image: nginx
imagePullPolicy: IfNotPresent
restartPolicy: Never

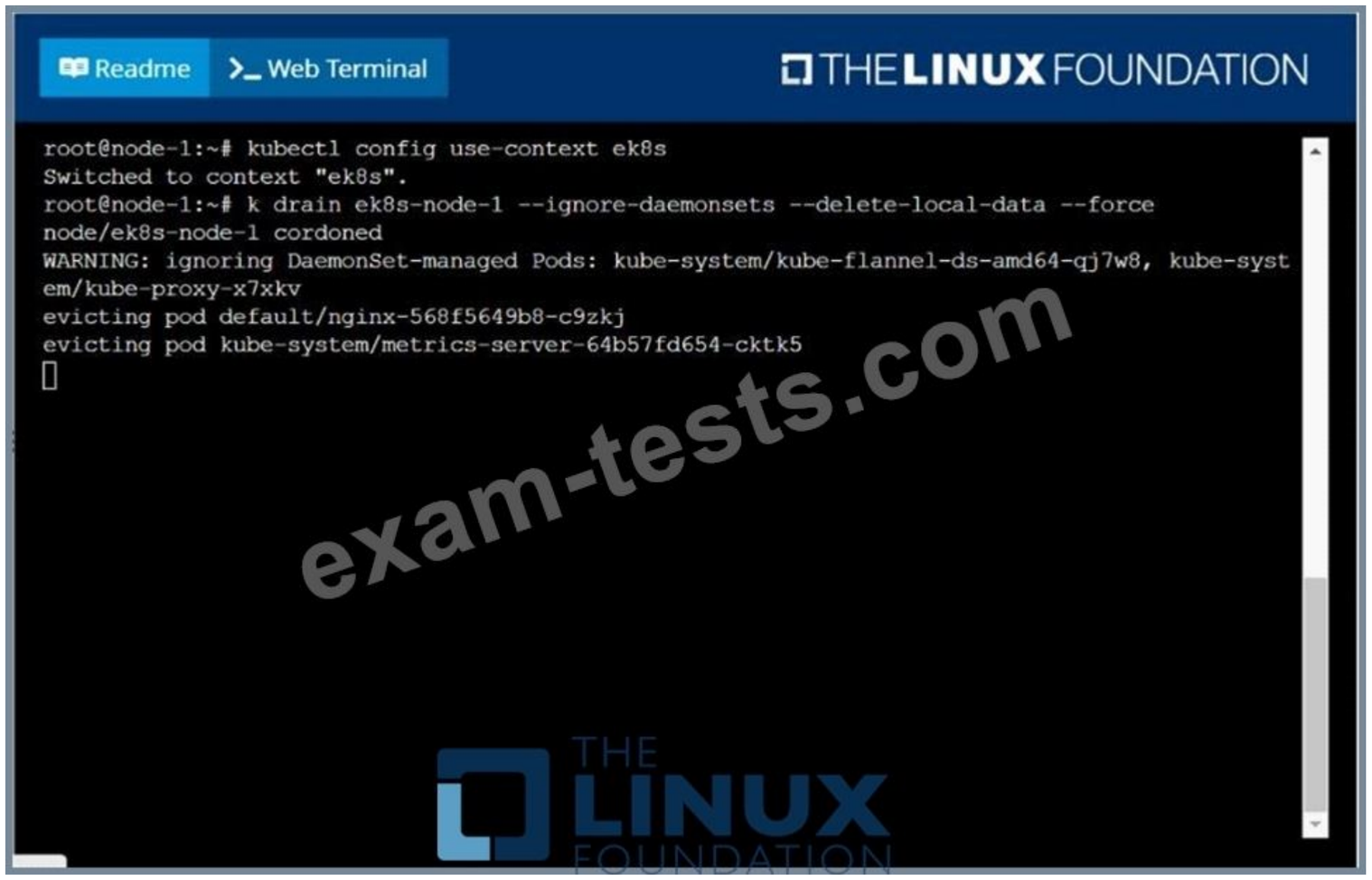
kubectl create -f nginx-pod.yaml
B. kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml > nginx-pod.yaml kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml | kubectl create -n engineering -f - YAML File:
apiVersion: v1
kind: Pod
metadata:
name: nginx
namespace: engineering
labels:
env: test
spec:
containers:
- name: nginx
image: nginx
imagePullPolicy: IfNotPresent
restartPolicy: Never
kubectl create -f nginx-pod.yaml
Answer: (SHOW ANSWER)

NEW QUESTION: 22
Create a pod that echo "hello world" and then exists. Have the pod deleted automatically when it's completed
Answer:
kubectl run busybox --image=busybox -it --rm --restart=Never --
/bin/sh -c 'echo hello world'
kubectl get po # You shouldn't see pod with the name "busybox"

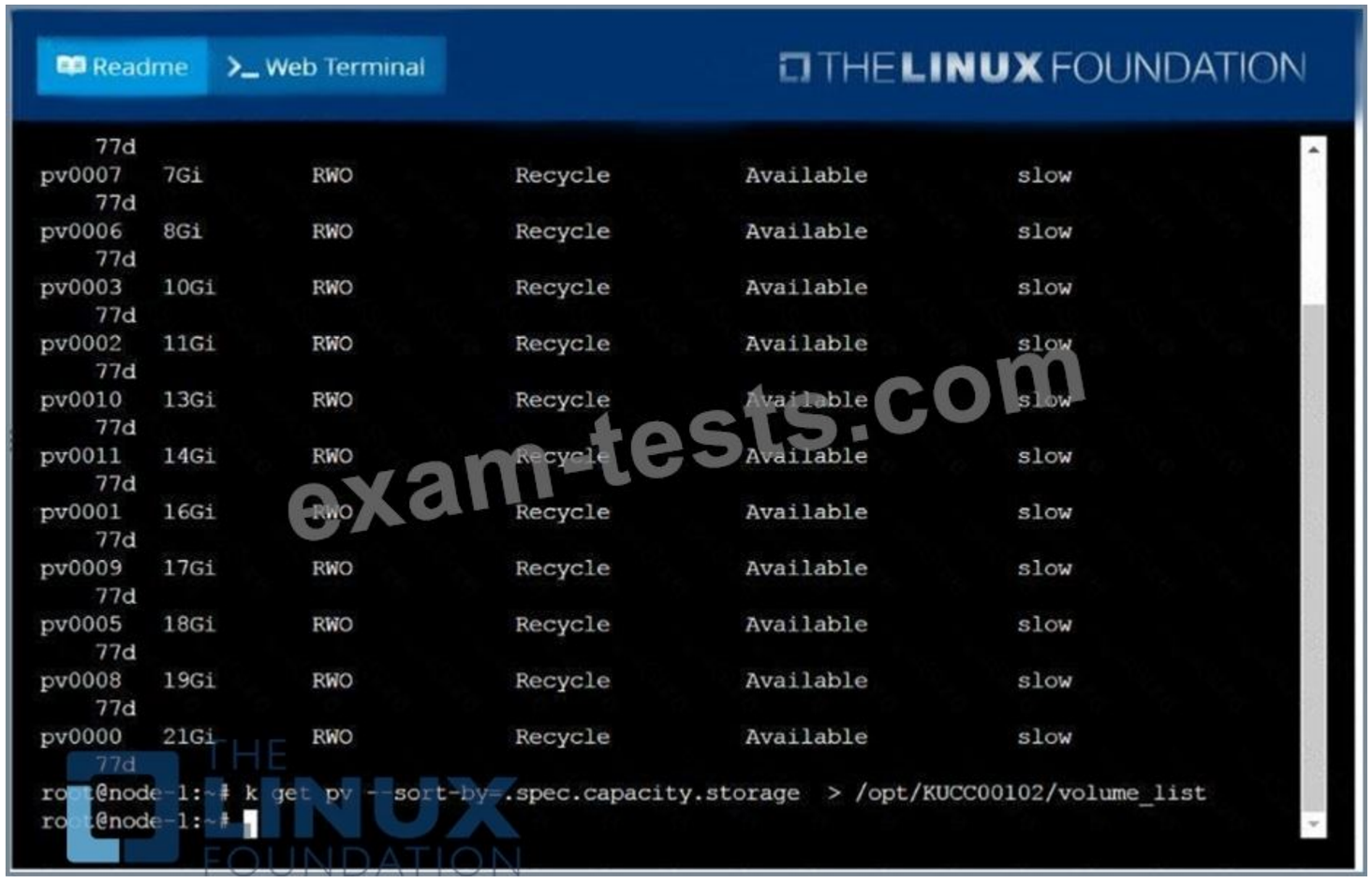
NEW QUESTION: 23
You must connect to the correct host.
Failure to do so may result in a zero score.
[candidate@base] $ ssh Cka000060
Task
Install Argo CD in the cluster by performing the following tasks:
Add the official Argo CD Helm repository with the name argo
The Argo CD CRDs have already been pre-installed in the cluster
Generate a template of the Argo CD Helm chart version 7.7.3 for the argocd namespace and save it to ~/argo- helm.yaml . Configure the chart to not install CRDs.
Answer:
Task Summary
* SSH into cka000060
* Add the Argo CD Helm repo named argo
* Generate a manifest (~/argo-helm.yaml) for Argo CD version 7.7.3
* Target namespace: argocd

* Do not install CRDs
* Just generate, don't install
# Step-by-Step Solution
1## SSH into the correct host
ssh cka000060
Required - skipping this = zero score
2## Add the Argo CD Helm repository
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update
This adds the official Argo Helm chart source.
3## Generate Argo CD Helm chart template (version 7.7.3)
Use the helm template command to generate a manifest and write it to ~/argo-helm.yaml.
helm template argocd argo/argo-cd \
--version 7.7.3 \
--namespace argocd \
--set crds.install=false \
> ~/argo-helm.yaml
* argocd # Release name (can be anything; here it's same as the namespace)
* --set crds.install=false # Disables CRD installation
* > ~/argo-helm.yaml # Save to required file
# 4## Verify the generated file (optional but smart)
head ~/argo-helm.yaml
Check that it contains valid Kubernetes YAML and does not include CRDs.
Final Command Summary
ssh cka000060
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update
helm template argocd argo/argo-cd \
--version 7.7.3 \
--namespace argocd \
--set crds.install=false \
> ~/argo-helm.yaml
head ~/argo-helm.yaml # Optional verification

NEW QUESTION: 24
Set the node named ek8s-node-1 as unavailable and reschedule all the pods running on it.
Answer:
solution



NEW QUESTION: 25
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.
Answer:
See the solution below.
Explanation
solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\2 C.JPG



NEW QUESTION: 26
Check the rollout history and make sure everything is ok after the update
A. kubectl rollout history deploy webapp
kubectl get deploy webapp --show-labels
kubectl get rs -
kubectl get po -l app=webapp
B. kubectl rollout history deploy webapp
kubectl get deploy webapp --show-labels
kubectl get rs -l app=webapp
kubectl get po -l app=webapp
Answer: (SHOW ANSWER)

NEW QUESTION: 27
You need to set up a load balancer for your Nginx service with the following requirements:
- Session affinity: Preserve client sessions across multiple pods, even if the pod is restarted or rescheduled.

- Health checks: Regularly check the health of Nginx pods and automatically remove unhealthy pods from the load balancer pool.
- Custom header: Add a custom header with the name "X-App-Version" and value "vl .0" to all requests to your Nginx service. How would you configure your Kubernetes resources to meet these requirements?

Answer:

See the solution below with Step by Step Explanation.

Explanation:

Solution (Step by Step) :

1. Define the Service:

- Create a Service of type "LoadBalancer" for your Nginx service.
- Include the sessionAffinity' field with a value of 'ClientIP' to enable client IP-based session affinity.
- Example:

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  sessionAffinity: ClientIP
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```


2. Configure the Deployment: - In your Nginx Deployment, define a liveness probe and readiness probe to check the health of your Nginx containers. - Example:

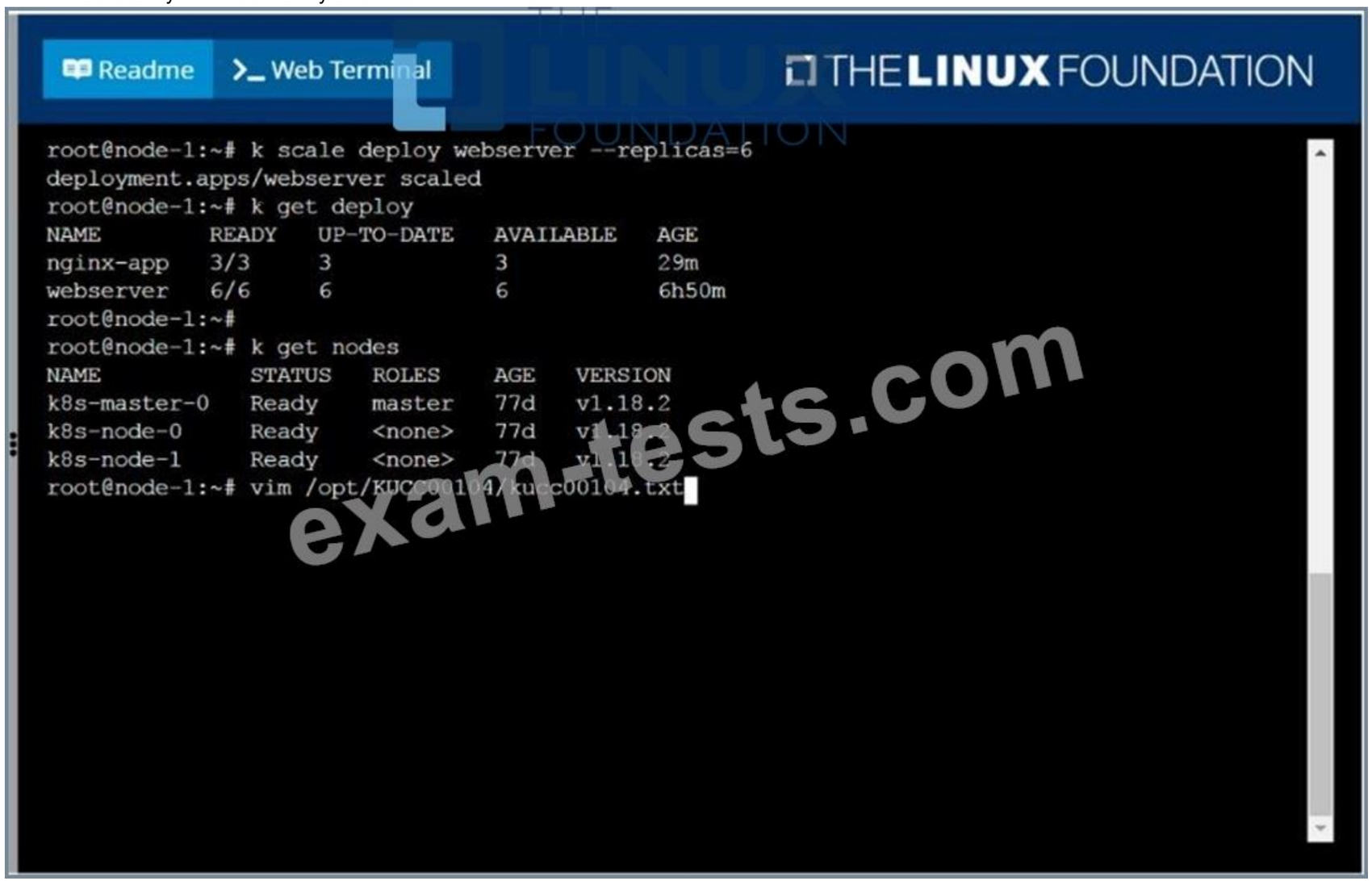
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        livenessProbe:
          tcpSocket:
            port: 80
          initialDelaySeconds: 15
          periodSeconds: 20
          failureThreshold: 3
        readinessProbe:
          tcpSocket:
            port: 80
          initialDelaySeconds: 5
          periodSeconds: 10
          failureThreshold: 2
```

3. Implement the Custom Header: - Configure an Ingress resource with the nginx.ingress.kubernetes.io/add-request-headeo annotation. - Example:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    nginx.ingress.kubernetes.io/add-request-header: "X-App-Version: v1.0"
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

4. Apply the Configurations: - Apply the updated Service, Deployment, and Ingress resources using 'kubectl apply -f service.yaml -f deployment.yaml -f ingress.yamr. 5. Verify the Load Balancer: - Access the Nginx service using the external IP address provided by the LoadBalancer. - Verify session affinity by making multiple requests and observing that they are consistently routed to the same pod. - Check the "X-App-Version" header in the responses to confirm that it is set to "vl .0".

NEW QUESTION: 28
Check to see how many worker nodes are ready (not including nodes tainted NoSchedule) and write the number to /opt/KUCC00104/kucc00104.txt.
Answer:
See the solution below.
Explanation
solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\15 B.JPG



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

NEW QUESTION: 29
From the pod labelname=cpu-utilizer, find podsrunning high CPU workloads and write the name of the pod consumingmost CPU to thefile/opt/KUTR00102/KUTR00102.txt(which already exists).
Answer:
See the solution below.
Explanation
solution


Readme
Web Terminal
THE LINUX FOUNDATION
root@node-1:~# k top po -l name=cpu-utilizer
NAME CPU(cores) MEMORY(bytes)
cpu-utilizer-98b9se 60m 7Mi
cpu-utilizer-ab2d3s 14m 7Mi
cpu-utilizer-kipb9a 45m 7Mi
root@node-1:~# vim /opt/KUTR00102/KUTR00102.txt
exam-tests.com
THE LINUX FOUNDATION

NEW QUESTION: 30
Given a partially-functioning Kubernetes 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.
You can ssh to the relevant I nodes (
[student@node-1] $ ssh <nodename
You can assume elevated privileges on any node in the cluster with the following command:
[student@nodename] $ | sudo -i
Answer:
See the solution below.
Explanation
solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\23 C.JPG

Readme | Web Terminal

THE LINUX FOUNDATION

```
root@node-1:~#
root@node-1:~# kubectl config use-context bk8s
Switched to context "bk8s".
root@node-1:~# ssh bk8s-master-0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

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

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

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

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

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


student@bk8s-master-0:~$ sudo -i
root@bk8s-master-0:~# vim /var/lib/kubelet/config.yaml
```

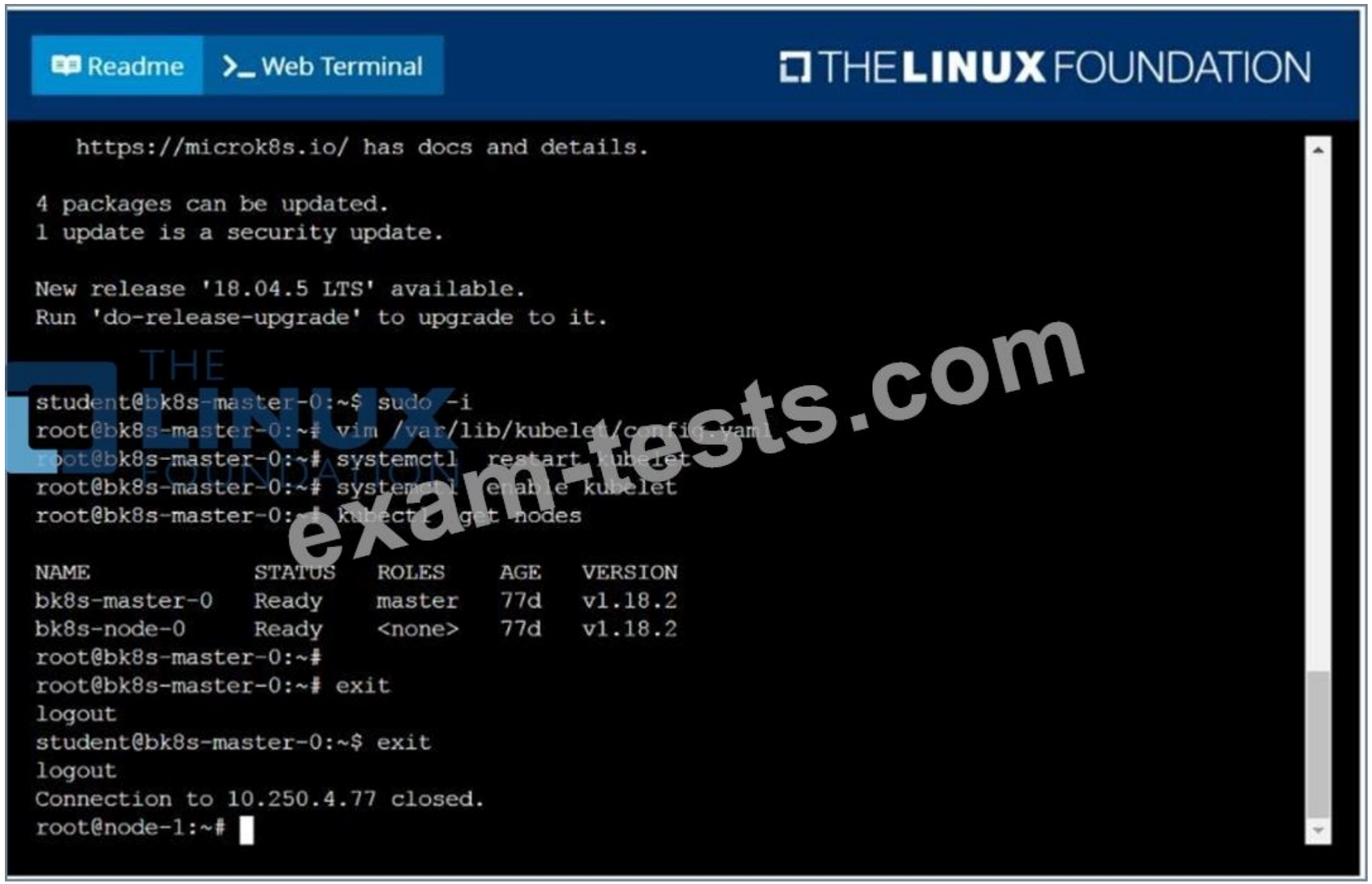

F:\Work\Data Entry Work\Data Entry\20200827\CKA\23 D.JPG

Readme | >_ Web Terminal

THE LINUX FOUNDATION

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


F:\Work\Data Entry Work\Data Entry\20200827\CKA\23 E.JPG



NEW QUESTION: 31
Label a node as app=test and verify
Answer:
kubectl label node node-name app=test // Verify kubectl get no -show-labels kubectl get no -l app=test

NEW QUESTION: 32
Get the memory and CPU usage of all the pods and find out top 3 pods which have the highest usage and put them into the cpuusage.txt file
A. // Get the top 3 pods
kubectl top pod --all-namespaces | sort --reverse --key 3 --
numeric | head -8
// putting into file
kubectl top pod --all-namespaces | sort --reverse --key 6 --
numeric | head -6 > cpu-usage.txt
// verify
cat cpu-usage.txt
B. // Get the top 3 pods
kubectl top pod --all-namespaces | sort --reverse --key 3 --
numeric | head -3
// putting into file
kubectl top pod --all-namespaces | sort --reverse --key 3 --
numeric | head -3 > cpu-usage.txt
// verify
cat cpu-usage.txt
Answer: B (LEAVE A REPLY)

NEW QUESTION: 33
You are tasked with setting up a Kubernetes cluster for a company with a strict security policy. They have two different teams: Development and Operations, with distinct access requirements.
The Development team needs read-only access to all resources and the ability to create and manage deployments, pods, and services. The Operations team needs full access to all resources, including cluster management, and the ability to perform actions like scaling, rolling updates, and troubleshooting.
Design a role-based access control (RBAC) configuration using Kubernetes resources (Role, RoleBinding, ClusterRole, ClusterRoleBinding) to grant appropriate access to each team.
Answer:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
Step 1: Create a Role for the Development team:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: dev-role
  namespace: default
rules:
- apiGroups: ["apps", "extensions", "v1"]
  resources: ["deployments", "pods", "services"]
  verbs: ["get", "list", "watch", "create", "update", "delete", "patch"]
- apiGroups: [""]
  resources: [""]
  verbs: ["get", "list", "watch"]
```

Step 2: Create a RoleBinding for the Development team:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-rolebinding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: dev-role
subjects:
- kind: User
  name: dev-user
  apiGroup: rbac.authorization.k8s.io
```

Step 3: Create a ClusterRole for the Operations team:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ops-clusterrole
rules:
- apiGroups: [""]
  resources: [""]
  verbs: [""]
- nonResourceURLs: [""]
  verbs: [""]
```

Step 4: Create a ClusterRoleBinding for the Operations team:

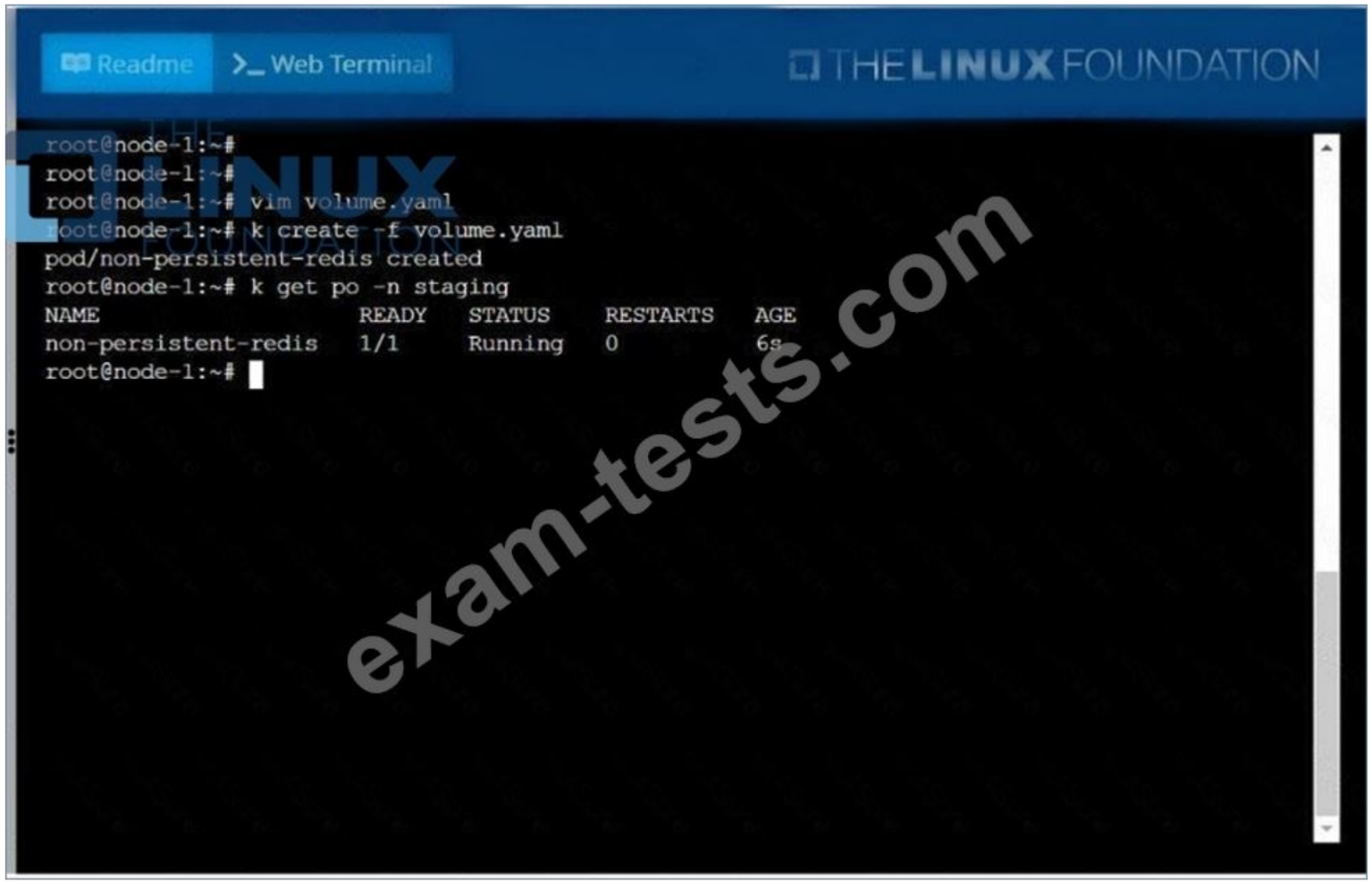
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: ops-clusterrolebinding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ops-clusterrole
subjects:
- kind: User
  name: ops-user
  apiGroup: rbac.authorization.k8s.io
```


The Role for the Development team grants specific permissions to create, manage, and view deployments, pods, and services. It also grants read-only access to all other resources. The RoleBinding links the "dev-role" to the "dev-user" account, allowing them to perform actions as defined by the Role. The ClusterRole for the Operations team provides full access to all cluster resources and allows them to perform any action, including cluster management tasks. The ClusterRoleBinding links the "ops-clusterrole" to the "ops-user" account, granting them complete control over the cluster. Note: You can replace "dev-user" and "ops-user" with actual user names or service accounts.

NEW QUESTION: 34
Create a pod as follows:
* Name: non-persistent-redis
* container Image: redis
* Volume with name: cache-control
* Mount path: /data/redis
The pod should launch in the staging namespace and the volume must not be persistent.
Answer:
See the solution below.
Explanation
solution


Readme
Web Terminal
THE LINUX FOUNDATION
root@node-1:~#
root@node-1:~#
root@node-1:~# vim volume.yaml
exam-tests.com
THE LINUX FOUNDATION


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



NEW QUESTION: 35
Create a NetworkPolicy which denies all ingress traffic
A. apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: default-deny
spec:
podSelector: {}
policyTypes:
- Ingress
B. apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: default-deny

spec:
podSelector: ()
policyTypes:
- Ingress
Answer: (SHOW ANSWER)

NEW QUESTION: 36
Create a pod that having 3 containers in it? (Multi-Container)
Answer:
image=nginx, image=redis, image=consul
Name nginx container as "nginx-container"
Name redis container as "redis-container"
Name consul container as "consul-container"
Create a pod manifest file for a container and append container
section for rest of the images
kubectl run multi-container --generator=run-pod/v1 --image=nginx --
dry-run -o yaml > multi-container.yaml
then
vim multi-container.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: multi-container
name: multi-container
spec:
containers:
- image: nginx
name: nginx-container
- image: redis
name: redis-container
- image: consul
name: consul-container
restartPolicy: Always

NEW QUESTION: 37
Task Weight: 4%

Task

Schedule a Pod as follows:

* Name: kucc1

* App Containers: 2

* Container Name/Images:

o nginx

o consul

Answer:

Solution:

```
student@node-1:~$ kubectl config use-context k8s
Switched to context "k8s".
student@node-1:~$ kubectl run kucc1   --image=nginx --dry-run=client -o yaml > aa.y
```

Readme | Web Terminal

THE LINUX FOUNDATION

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: kucc1
  name: kucc1
spec:
  containers:
  - image: nginx
    name: nginx
  - image: consul
    name: consul
```

NEW QUESTION: 38

You are managing a Kubernetes cluster with a complex deployment scenario. The cluster has multiple namespaces, each with its own set of applications and users. You need to create a robust RBAC system to enforce fine-grained access control.

Current Setup:

Namespace: 'dev', 'staging', 'production'

Users: 'developer', 'qa', 'admin'

Applications: 'appl', 'app2' in 'dev', 'app3' in 'staging', 'app4' in 'production' Requirements:

'developer' should be able to access and manage 'appl' and 'app2' in the 'dev' namespace.

'qa' should be able to access and manage 'app3' in the 'staging' namespace.

'admin' should have full cluster-wide access.

Task:

Create the necessary Role, RoleBinding, and ClusterRole objects to implement this RBAC system.

Answer:

See the solution below with Step by Step Explanation.

Explanation:

Solution (Step by Step) :

1. Create Roles for 'developer' and 'qa':

```
kubectl create role developer-role --namespace=dev --verb=get,list,watch,create,update,patch,delete --resource=pods,services,deployments --resource-names=app1,app2
```

```
kubectl create role qa-role --namespace=staging --verb=get,list,watch,create,update,patch,delete --resource=pods,services,deployments --resource-names=app3
```

2. Create RoleBinding for 'developer' and 'qa':

```
kubectl create rolebinding developer-rolebinding --namespace=dev --role=developer-role --user=developer
kubectl create rolebinding qa-rolebinding --namespace=staging --role=qa-role --user=qa
```

3. Create ClusterRole for 'admin'.

```
kubectl create clusterrole admin-clusterrole --verb=get,list,watch,create,update,patch,delete --resource= --resource-names=
```

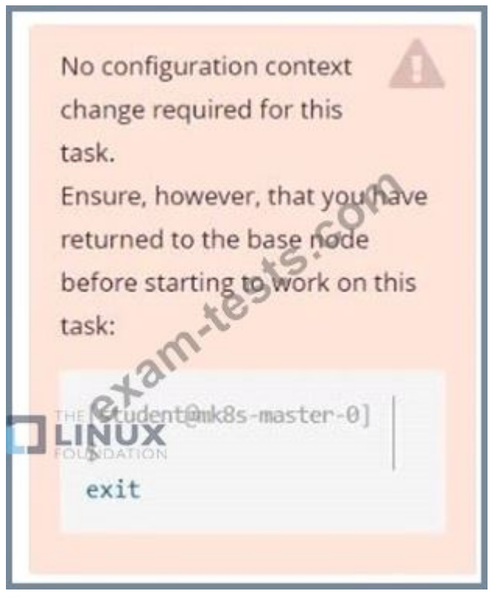
4. Create ClusterRoleBindin for 'admin'.

```
kubectl create clusterrolebinding admin-clusterrolebinding --clusterrole=admin-clusterrole --user= admin
```

We created separate roles (developer-role', 'cp-role') for each user group, limiting their access to specific namespaces and resources. We bound these roles to users using RoleBindings in the respective namespaces. For 'admin', we created a ClusterRole Cadmin-clusterrole') with full access to all resources, and bound it using a ClusterRoleBinding. This setup ensures that each user has appropriate access rights based on their role and responsibilities. ,

NEW QUESTION: 39

Score: 7%



Task

First, create a snapshot of the existing etcd instance running at https://127.0.0.1:2379, saving the snapshot to /srv/data/etcd-snapshot.db.

Next, restore an existing, previous snapshot located at /var/lib/backup/etcd-snapshot-previo us.db

Answer:

Solution:

#backup

ETCDCTL_API=3 etcdctl --endpoints="https://127.0.0.1:2379" --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key snapshot save /etc/data/etcd-snapshot.db

#restore

ETCDCTL_API=3 etcdctl --endpoints="https://127.0.0.1:2379" --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key snapshot restore /var/lib/backup/etcd-snapshot-previoys.db

NEW QUESTION: 40

You are running an application in Kubernetes using a Deployment that defines 3 replicas. You need to perform a rolling update to the Deployment to upgrade the application to a new version. During the update process, you want to ensure that at least 2 replicas are always available, and the maximum number of new pods that can be created at the same time is also limited to 1. How can you configure the Deployment to achieve this rolling update strategy?

Answer:

See the solution below with Step by Step Explanation.

Explanation:

Solution (Step by Step) :

1. Update Deployment YAML:

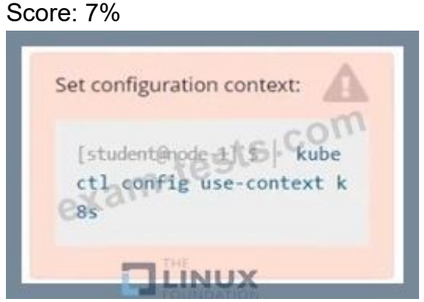
- Update the 'spec.replicas' field to the desired number of replicas for the new version.
- In the 'spec.strategy.rollingUpdate' section, set the 'maxUnavailable' to 1, meaning that only one pod can be unavailable during the update process.
- Set the maxSurge' to 1, limiting the number of new pods that can be created simultaneously to 1.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: my-app-image:latest
        imagePullPolicy: Always
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
```

2. Apply the Updated Deployment: - Use 'kubectl apply -f deployment.yaml' to apply the changes to your cluster. 3. Monitor the Update Process: - Use 'kubectl get pods -l app=my-app' to monitor the pods. You will see a rolling update in progress: - One old pod will be terminated at a time. - One new pod will be created at a time. - The update will continue until all replicas are updated to the new version. 4. Verify the Update: - Once the update is complete, use 'kubectl describe deployment my-deployment' to check the deployment status. The 'updatedReplicas' field should match the 'replicas' field, indicating that the update was successful. By using 'maxUnavailable' and 'maxSurge' you control the number of unavailable and surge pods during the update process. This ensures a safe and controlled rolling update strategy.,

NEW QUESTION: 41

Score: 7%



Task

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

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

Configure the new service to also expose the individual Pods via a NodePort on the nodes on which they are scheduled.

Answer:

See the solution below.

Explanation

Solution:

kubectl get deploy front-end

kubectl edit deploy front-end -o yaml

#port specification named http

#service.yaml

apiVersion: v1

kind: Service

metadata:

name: front-end-svc

labels:

app: nginx

spec:

ports:

- port: 80

protocol: tcp

name: http

selector:

app: nginx

type: NodePort

kubectl create -f service.yaml

kubectl get svc

port specification named http

kubectl expose deployment front-end --name=front-end-svc --port=80 --tarport=80 --type=NodePort

NEW QUESTION: 42

You have a Kubernetes cluster with a Deployment named 'web-app' that runs a web application. You need to set up a mechanism to automatically scale the deployment based on the CPU utilization of the pods. The scaling should be triggered when the average CPU utilization across all pods reaches 70%. You should set the minimum and maximum replicas to 2 and 5 respectively.

Answer:

See the solution below with Step by Step Explanation.

Explanation:

Solution (Step by Step) :

1. Create a Horizontal Pod Autoscaler (HPA):

- Use "kubectl create hpa' command to create an HPA resource.

- Specify the name of the HPA, the Deployment to scale, the target CPU utilization (70%), and the minimum and maximum replicas.

kubectl create hpa web-app-hpa --min=2 --max=5 --cpu-utilization-percentage=70 --target- ref=Deployment/web-app

2. Verify the HPA Creation:

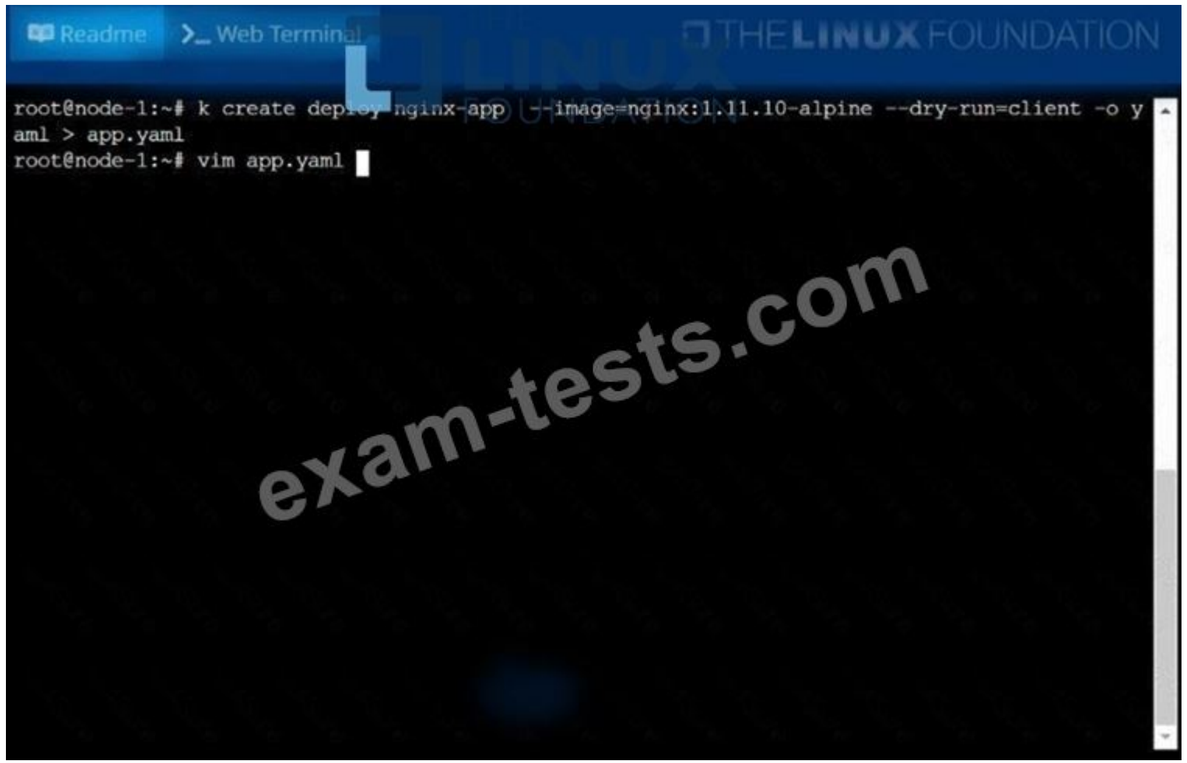
- Use 'kubectl get hpa' command to check if the HPA was created successfully. You should see an HPA named 'web-app-hpa' with the configured settings.

3. Monitor the Scaling Behavior:
- You can use the 'kubectl get pods -l command to monitor the number of pods running as the CPU utilization changes.
- When the average CPU utilization across the pods reaches 70%, the HPA will automatically scale up the Deployment to add more pods.
- Conversely, when the CPU utilization falls below the threshold, the HPA will scale down the Deployment to reduce the number of pods.
Ensure that the 'metrics-server' is installed in your cluster to enable CPU utilization monitoring.,

NEW QUESTION: 43
You must connect to the correct host.
Failure to do so may result in a zero score.
[candidate@base] $ ssh Cka000051
Context
You manage a WordPress application. Some Pods are not starting because resource requests are too high.
Your task Is to prepare a Linux system for Kubernetes . Docker is already installed, but you need to configure it for kubeadm .
Task
Complete these tasks to prepare the system for Kubernetes :
Set up cri-dockerd :
. Install the Debian package
~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_am
d64.deb
Debian packages are installed using
dpkg .
. Enable and start the cri-docker service
Configure these system parameters:
. Set net.bridge.bridge-nf-call-iptables to 1
Answer:
Task Summary
You are given a host to prepare for Kubernetes:
* Use dpkg to install cri-dockerd
* Enable and start the cri-docker service
* Set net.bridge.bridge-nf-call-iptables to 1 via sysctl
Step-by-Step Instructions
1## SSH into the correct node
bash
CopyEdit
ssh cka000051
Required - failure to connect to the correct host = zero score.
2## Install cri-dockerd
You are told the .deb file is already located at:
bash
CopyEdit
~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb

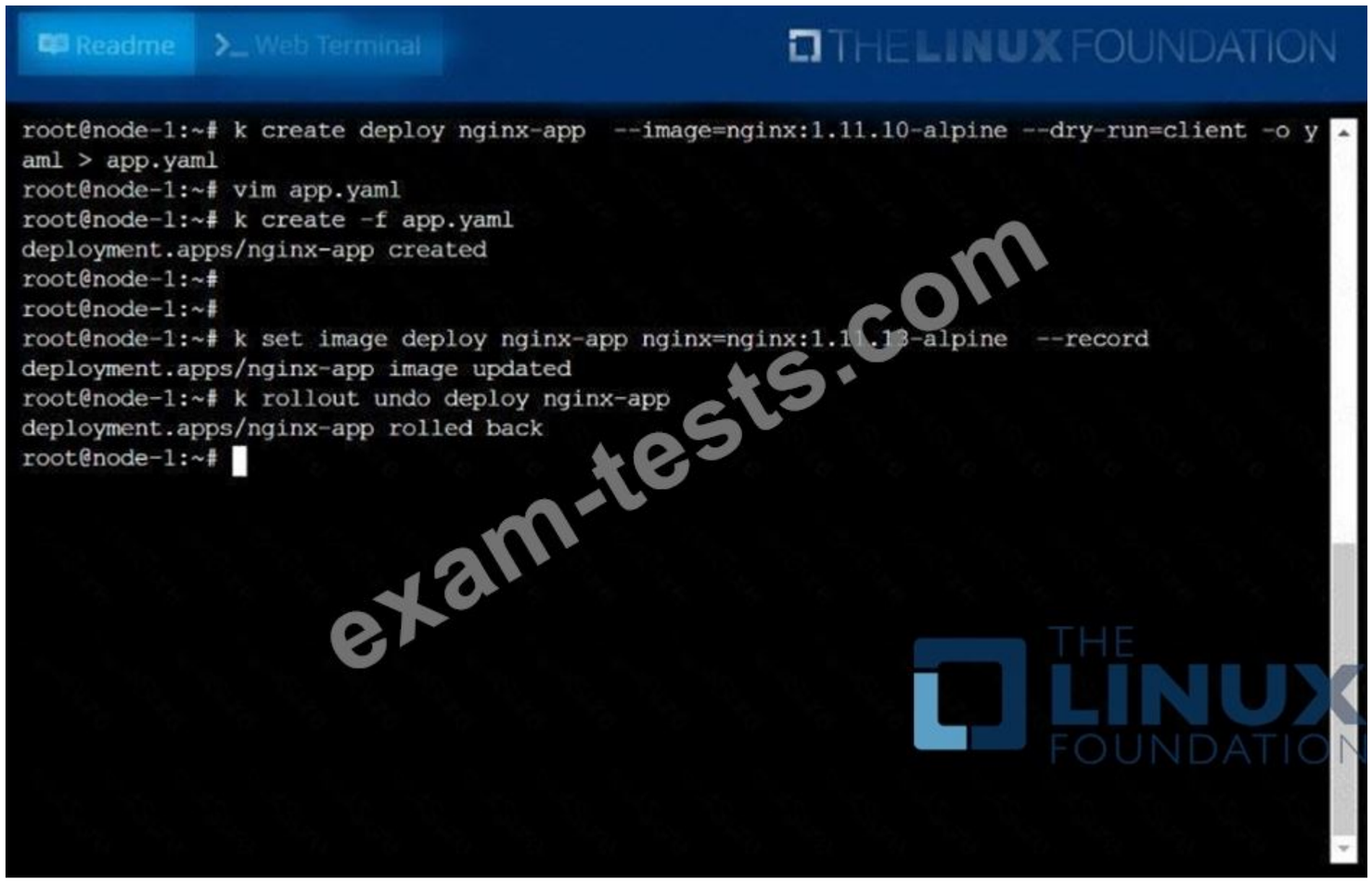
Install it with dpkg:
bash
CopyEdit
sudo dpkg -i ~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb
# If any dependencies are missing (e.g., golang or containerd), you might need:
bash
CopyEdit
sudo apt-get install -f -y
But usually, the exam system provides a pre-validated .deb environment.
3## Enable and start cri-docker service
Start and enable both services:
bash
CopyEdit
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket
sudo systemctl start cri-docker.service
Check status (optional but smart):
bash
CopyEdit
sudo systemctl status cri-docker.service
You should see it active (running).
4## Configure the sysctl parameter
Set net.bridge.bridge-nf-call-iptables=1 immediately and persistently.
Step A: Apply immediately:
sudo sysctl net.bridge.bridge-nf-call-iptables=1
Step B: Persist it in /etc/sysctl.d:
Create or modify a file:
echo "net.bridge.bridge-nf-call-iptables = 1" | sudo tee /etc/sysctl.d/k8s.conf Reload sysctl:
sudo sysctl --system
Verify:
sysctl net.bridge.bridge-nf-call-iptables
Should return:
net.bridge.bridge-nf-call-iptables = 1
Now the system is ready for kubeadm with Docker (via cri-dockerd)!
ssh cka000051
sudo dpkg -i ~/cri-dockerd_0.3.9.3-0.ubuntu-jammy_amd64.deb
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket
sudo systemctl start cri-docker.service
sudo sysctl net.bridge.bridge-nf-call-iptables=1
echo "net.bridge.bridge-nf-call-iptables = 1" | sudo tee /etc/sysctl.d/k8s.conf sudo sysctl --system

NEW QUESTION: 44
Create a deployment as follows:
* Name:nginx-app
* Using containernginxwithversion 1.11.10-alpine
* The deployment should contain3replicas
Next, deploy the application with newversion1.11.13-alpine, byperforming a rolling update.
Finally, rollback that update to theprevious version1.11.10-alpine.
Answer:
See the solution below.
Explanation
solution



Readme Web Terminal

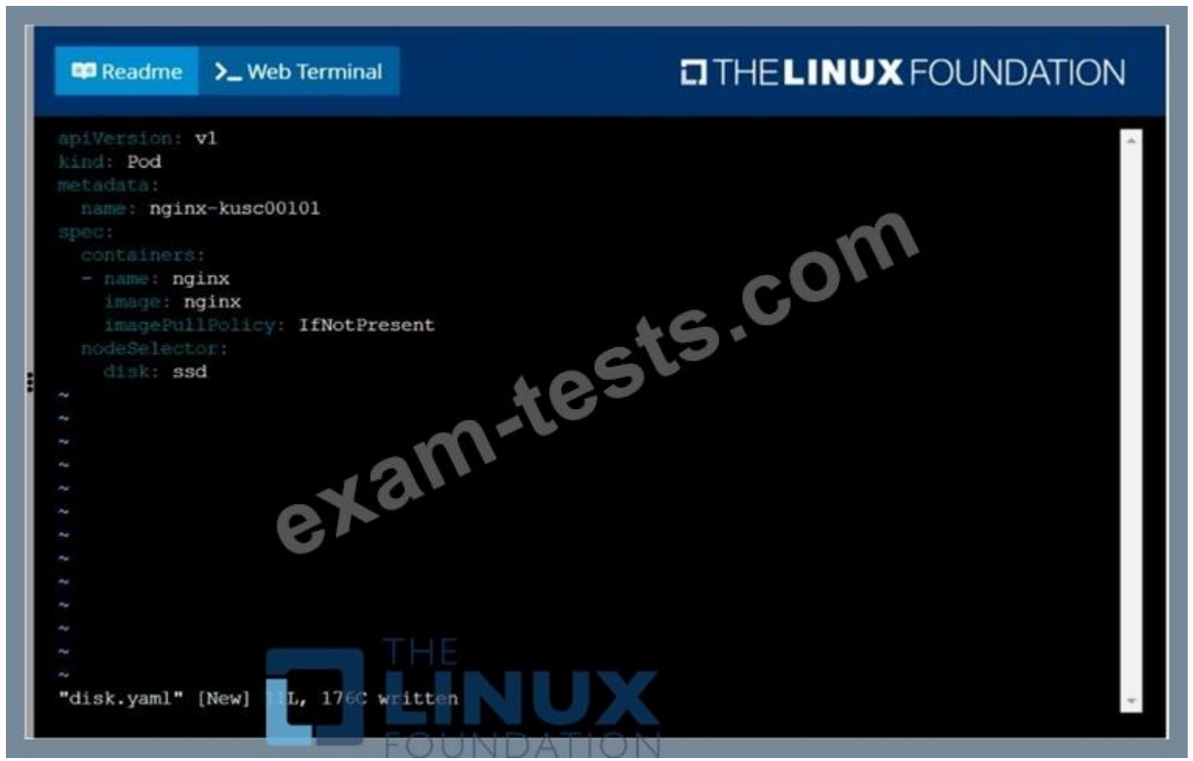
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - image: nginx:1.11.10-alpine
        name: nginx
~
~
~
~
~
~
~
~
"app.yaml
```



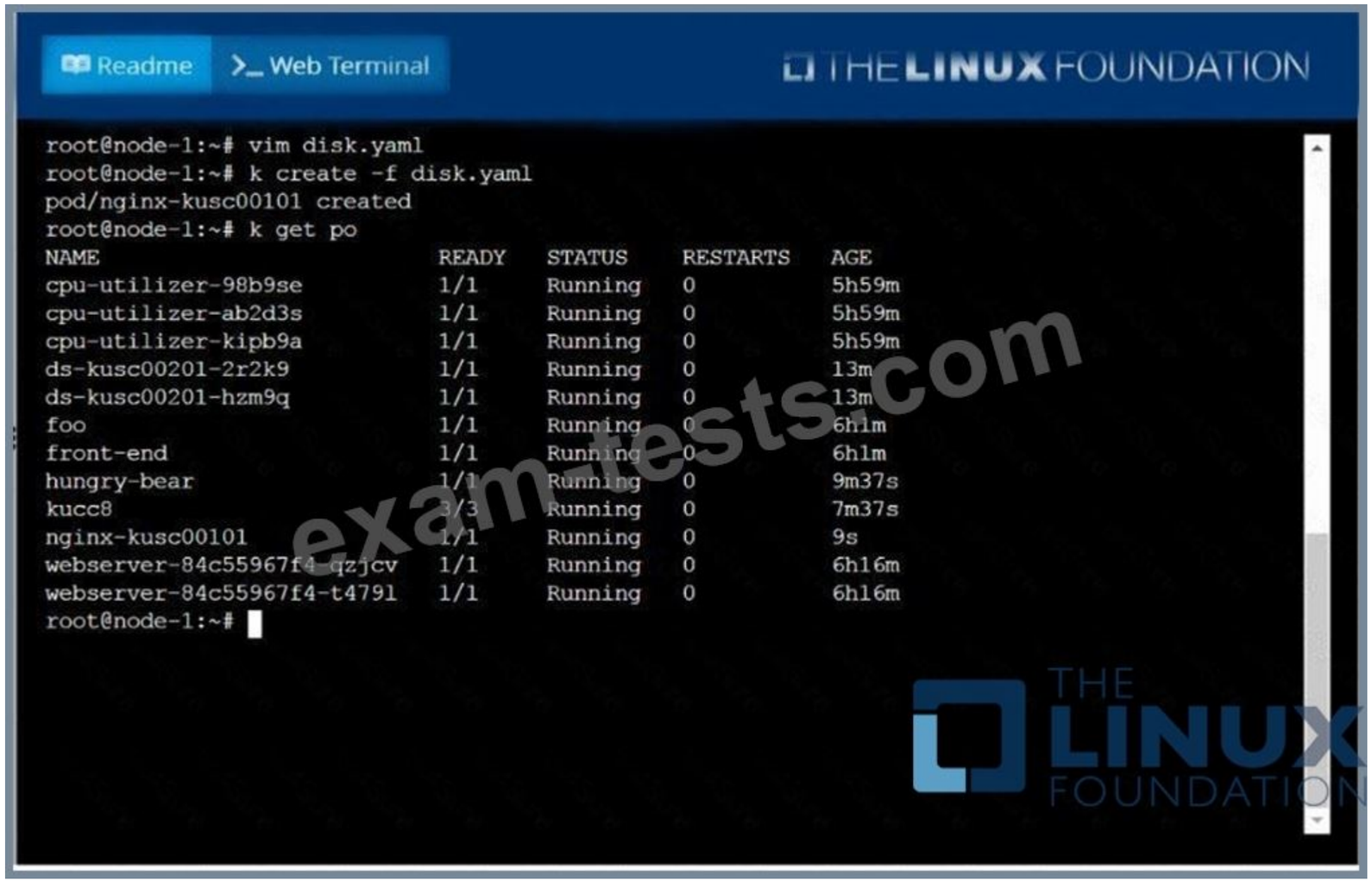
NEW QUESTION: 45
Create a pod that echo "hello world" and then exists. Have the pod deleted automatically when it's completed
Answer:
kubectl run busybox --image=busybox -it --rm --restart=Never -- /bin/sh -c 'echo hello world' kubectl get po # You shouldn't see pod with the name "busybox"

NEW QUESTION: 46
Schedule a pod as follows:
Name: nginx-kusc00101
Image: nginx
Node selector: disk=ssd
Answer:
See the solution below.
Explanation
solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\6 B.JPG

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



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



NEW QUESTION: 47
You have an application deployed in your Kubernetes cluster that is configured to access a private database hosted on another Kubernetes cluster in a different VPC. How would you configure a secure and efficient way for your application to communicate with the private database?
Answer:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :

1 . Create a Service Account in the Database Cluster:

- Create a Service Account in the database cluster that will be used by the application in the other cluster.

- Example:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: database-client
```

- Apply this configuration using 'kubectl apply -f service-account.yaml' in the database cluster. 2. Generate a Secret for the Service Account: - Generate a secret for the Service Account. This secret will contain the credentials needed to access the database. - Example: bash kubectl create secret generic database-client-secret --from-literal=username= -n - Replace and with your actual credentials and " with the namespace where your database service is running. 3. Configure a Network Policy: - Create a Network Policy in the database cluster to allow traffic from your application's pods in the other cluster. - Example:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-app-access
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app:
```

- Replace with the label assigned to your application pods. 4. Create a Service in the Database Cluster: - Create a Service in the database cluster that exposes your database. This service will be used by your application to connect to the database. - Example:

```
apiVersion: v1
kind: Service
metadata:
  name: database-service
spec:
  ports:
  - port: 5432
    targetPort: 5432
    protocol: TCP
  selector:
    app: database
```

- Replace 'database' with the label of your database pods. 6. Configure the Application Deployment: - In your application's deployment configuration, mount the secret containing the database credentials and configure your application to connect to the database service in the other cluster using the service's DNS name. - Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 2
  template:
    spec:
      containers:
      - name: my-app
        image: my-app:latest
        env:
        - name: DATABASE_URL
          value: postgres://database-client:database-client-secret@database-service.database-namespace.svc.cluster.local:5432/your-database
        volumeMounts:
        - name: database-credentials
          mountPath: /var/secrets/database
      volumes:
      - name: database-credentials
        secret:
          secretName: database-client-secret
```


- Ensure that the application has the necessary network connectivity to access the database cluster through your network configuration. - Example: - Configure a VPC peering connection between the two VPCs to enable direct communication between the clusters. - Create a VPN connection between the clusters using a VPN gateway. 7. Test the Connection: - Verify that your application can successfully connect to and interact with the database.

NEW QUESTION: 48
List "nginx-dev" and "nginx-prod" pod and delete those pods
Answer:
kubect1 get pods -o wide
kubectl delete po "nginx-dev" kubectl delete po "nginx-prod"

NEW QUESTION: 49
List all the pods sorted by created timestamp
Answer:
See the solution below.
Explanation
kubect1 get pods--sort-by=.metadata.creationTimestamp

NEW QUESTION: 50
You are setting up RBAC for a Kubernetes cluster with three namespaces: "development", "staging", and "production". You need to create a role binding that allows developers in the "development" namespace to create deployments, pods, and services, but only within their own namespace.

Answer:

See the solution below with Step by Step Explanation.

Explanation:

Solution (Step by Step) :

1 Create a Role for Developers:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: developer-role
  namespace: development
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["create", "get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["statefulsets"]
  verbs: ["create", "get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["daemonsets"]
  verbs: ["create", "get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["replicasets"]
  verbs: ["create", "get", "list", "watch"]
- apiGroups: ["core"]
  resources: ["pods"]
  verbs: ["create", "get", "list", "watch", "delete"]
- apiGroups: ["core"]
  resources: ["services"]
  verbs: ["create", "get", "list", "watch", "delete"]
- apiGroups: ["networking.k8s.io"]
  resources: ["ingresses"]
  verbs: ["create", "get", "list", "watch"]
```

2. Create a Role Binding for Developers:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: developer-rolebinding
  namespace: development
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-role
subjects:
- kind: User
  name: developer-user
  apiGroup: rbac.authorization.k8s.io
```

3. Replace 'developer-user' with the actual username of a developer. 4. Repeat steps 1-3 for other namespaces ("staging" and "production") with appropriate resource permissions and user names. 5. Verify the RBAC configuration: - Check that the roles and rolebindings are created successfully using 'kubectl get roles' and 'kubectl get rolebindings'. - Test the permissions by creating a pod, deployment, or service as a developer user in their respective namespaces. 6. If you encounter issues, review the configuration carefully, making sure the namespaces, roles, role bindings, and user names are correct.

NEW QUESTION: 51
Change the label for one of the pod to env=uat and list all the pods to verify
Answer:
kubectl label pod/nginx-dev3 env=uat --overwrite kubectl get pods --show-labels

NEW QUESTION: 52
You are running a Kubernetes cluster with a large number of deployments and services. You need to improve the performance and efficiency of DNS resolution, especially during peak traffic periods.
Answer:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Increase CoreDNS Resources:
- Allocate more CPU, memory, and storage resources to the CoreDNS Deployment to handle increased DNS traffic.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: coredns
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: coredns
        resources:
          requests:
            cpu: 2
            memory: 2Gi
          limits:
            cpu: 4
            memory: 4Gi
```

2. Configure CoreDNS for Efficient Caching: - Use CoreDNS's 'cache' plugin to store DNS records in memory and reduce the need for frequent DNS queries.

```
.:53 {
    errors
    health
    ready
    kubernetes cluster.local in-addr.arpa ip6.arpa {
        pods insecure
        fallthrough
    }
    # Configure caching with a longer TTL
    cache 600
    reload 10s
}
```

3. Use a Distributed DNS Server: - If you have a very large cluster with high traffic, consider using a distributed DNS server like etcd or Consul. This can help to improve performance and scalability. 4. Use DNS over TLS (DOT) or DNS over HTTPS (DoH): - Enable secure DNS communication to reduce the risk of DNS poisoning attacks, which can significantly impact performance.

```
.:53 {
    errors
    health
    ready
    # Use DoH to query Cloudflare DNS
    forward . https://1.1.1.1:853 {
        max_tcp_size 4096
        max_udp_size 4096
    }
    cache 30
    reload 10s
}
```


5. Monitor CoreDNS Performance: - Use metrics and logs to monitor CoreDNS performance and identify potential bottlenecks. This will help you adjust your configuration and resource allocation as needed.]

NEW QUESTION: 53
How can an administrator configure the NGFW to automatically quarantine a device using Global Protect?
A. by exporting the list of quarantined devices to a pdf or csv file by selecting PDF/CSV at the bottom of the Device Quarantine page and leveraging the appropriate XSOAR playbook.
B. There is no native auto-quarantine feature so a custom script would need to be leveraged.
C. by using security policies, log forwarding profiles, and log settings.
D. by adding the device's Host ID to a quarantine list and configure GlobalProtect to prevent users from connecting to the GlobalProtect gateway from a quarantined device.
Answer: D (LEAVE A REPLY)

NEW QUESTION: 54
Create a secret mysecret with values user=myuser and password=mypassword
A. kubectl create secret generic my-secret --fromliteral=username=user --from-literal=password=mypassword
// Verify
kubectl get secret generic my-secret -o yaml
B. kubectl create secret generic my-secret --fromliteral=username=user --from-literal=password=mypassword
// Verify
kubectl get secret --all-namespaces
kubectl get secret generic my-secret -o yaml
Answer: B (LEAVE A REPLY)

NEW QUESTION: 55
Score:7%

Context

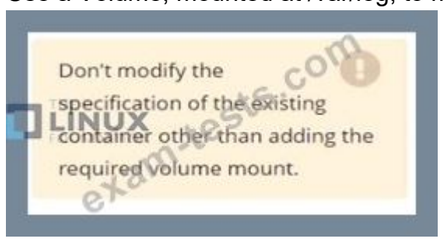
An existing Pod needs to be integrated into the Kubernetes built-in logging architecture (e. g. kubectl logs). Adding a streaming sidecar container is a good and common way to accomplish this requirement.

Task

Add a sidecar container named sidecar, using the busybox Image, to the existing Pod big-corp-app. The new sidecar container has to run the following command:

/bin/sh -c tail -n+1 -f /va r/log/big-corp-app.log

Use a Volume, mounted at /var/log, to make the log file big-corp-app.log available to the sidecar container.



Answer:

Solution:

```
#
kubectl get pod big-corp-app -o yaml
#
apiVersion: v1
kind: Pod
metadata:
name: big-corp-app
spec:
containers:
- name: big-corp-app
image: busybox
args:
- /bin/sh
- -c
- >
i=0;
while true;
do
echo "$(date) INFO $i" >> /var/log/big-corp-app.log;
i=$((i+1));
```

sleep 1;
done
volumeMounts:
- name: logs
mountPath: /var/log
- name: count-log-1
image: busybox
args: [/bin/sh, -c, 'tail -n+1 -f /var/log/big-corp-app.log']
volumeMounts:
- name: logs
mountPath: /var/log
volumes:
- name: logs
emptyDir: {
}
#
kubectl logs big-corp-app -c count-log-1

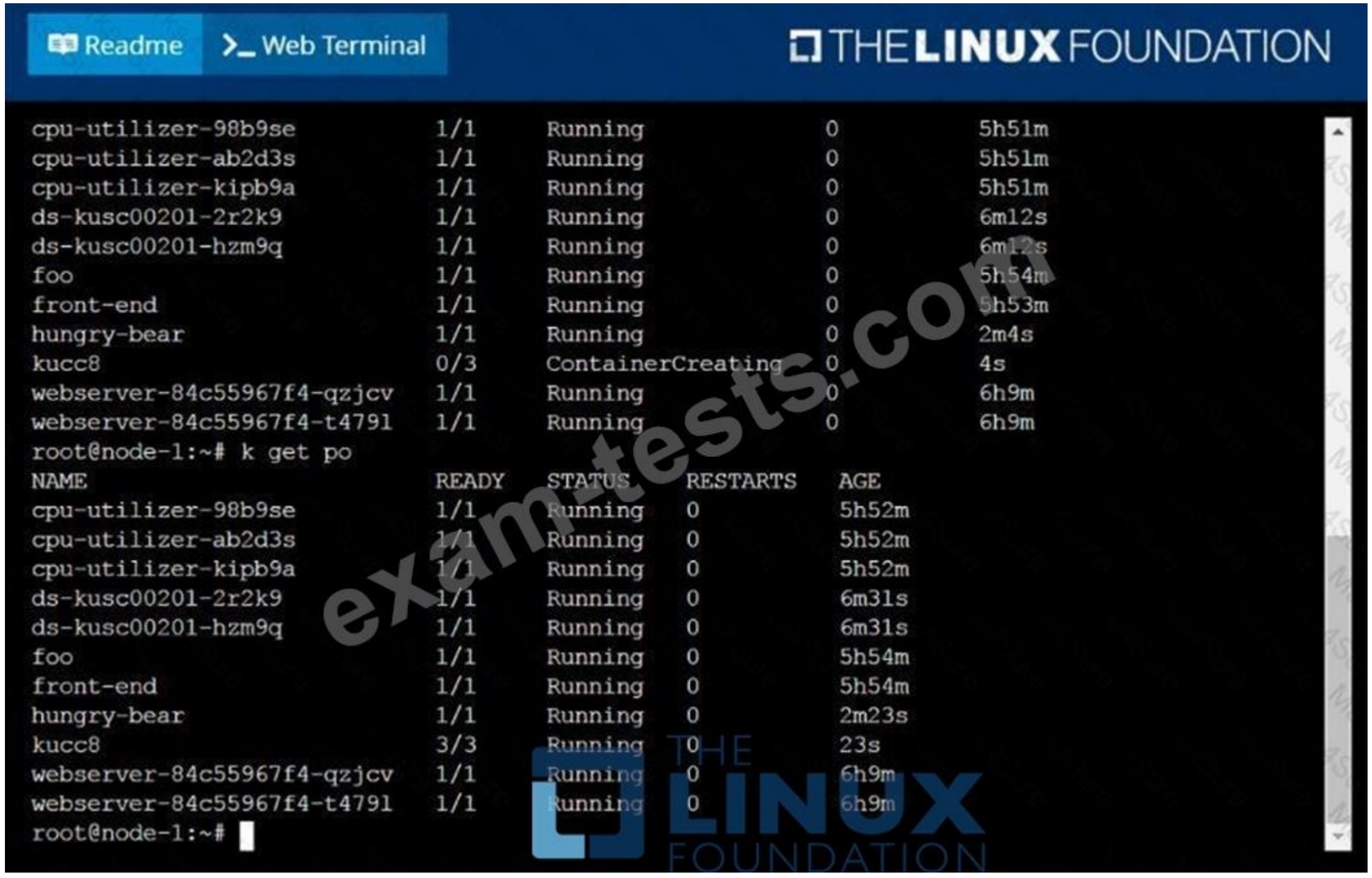
NEW QUESTION: 56
Create a pod named kucc8 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified):
nginx + redis + memcached.
Answer:


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

Readme | Web Terminal

THE LINUX FOUNDATION

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



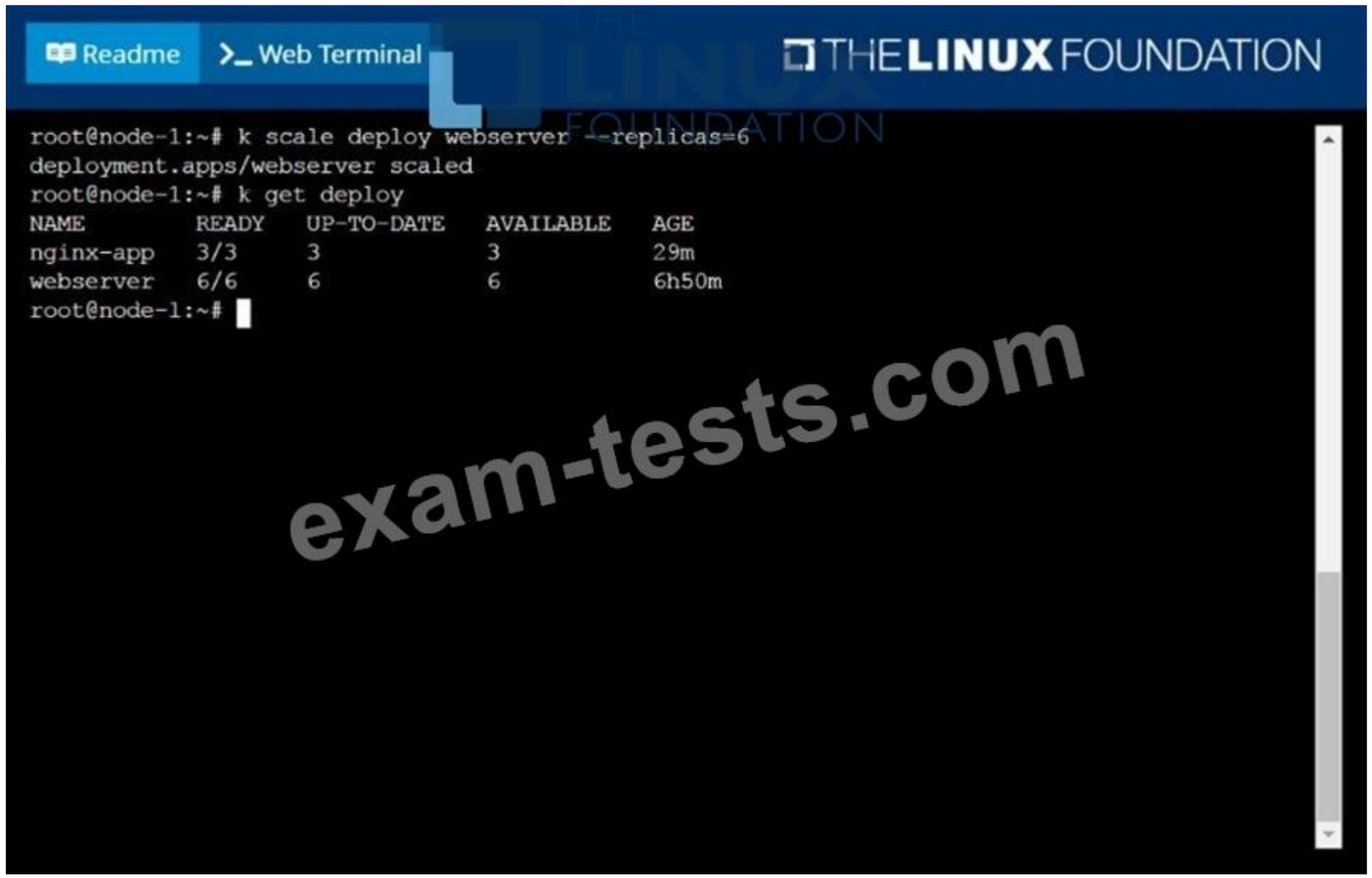
NEW QUESTION: 57
You must connect to the correct host.
Failure to do so may result in a zero score.
[candidate@base] $ ssh Cka000047
Task
A MariaDB Deployment in the mariadb namespace has been deleted by mistake. Your task is to restore the Deployment ensuring data persistence. Follow these steps:
Create a PersistentVolumeClaim (PVC) named mariadb in the mariadb namespace with the following specifications:
Access mode ReadWriteOnce
Storage 250Mi
You must use the existing retained PersistentVolume (PV).
Failure to do so will result in a reduced score.
There is only one existing PersistentVolume .

Edit the MariaDB Deployment file located at ~/mariadb-deployment.yaml to use PVC you created in the previous step.
Apply the updated Deployment file to the cluster.
Ensure the MariaDB Deployment is running and stable.
Answer:
Task Overview
You're restoring a MariaDB deployment in the mariadb namespace with persistent data.
Tasks:
* SSH into cka000047
* Create a PVC named mariadb:
* Namespace: mariadb
* Access mode: ReadWriteOnce
* Storage: 250Mi
* Use the existing retained PV (there's only one)
* Edit ~/mariadb-deployment.yaml to use the PVC
* Apply the deployment
* Verify MariaDB is running and stable
Step-by-Step Solution
1## SSH into the correct host
ssh cka000047
Required - skipping = zero score
2## Inspect the existing PersistentVolume
kubectl get pv
Identify the only existing PV, e.g.:
NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS
mariadb-pv 250Mi RWO Retain Available <none> manual
Ensure the status is Available, and it is not already bound to a claim.
3## Create the PVC to bind the retained PV
Create a file mariadb-pvc.yaml:
cat <<EOF > mariadb-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: mariadb
namespace: mariadb
spec:
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 250Mi
volumeName: mariadb-pv # Match the PV name exactly

EOF
Apply the PVC:
kubectl apply -f mariadb-pvc.yaml
This binds the PVC to the retained PV.
4## Edit the MariaDB Deployment YAML
Open the file:
nano ~/mariadb-deployment.yaml
Look under the spec.template.spec.containers.volumeMounts and spec.template.spec.volumes sections and update them like so:
Add this under the container:
yaml
CopyEdit
volumeMounts:
- name: mariadb-storage
mountPath: /var/lib/mysql
And under the pod spec:
volumes:
- name: mariadb-storage
persistentVolumeClaim:
claimName: mariadb
These lines mount the PVC at the MariaDB data directory.
5## Apply the updated Deployment
kubectl apply -f ~/mariadb-deployment.yaml
6## Verify the Deployment is running and stable
kubectl get pods -n mariadb
kubectl describe pod -n mariadb <mariadb-pod-name>
Ensure the pod is in Running state and volume is mounted.
Final Command Summary
ssh cka000047
kubectl get pv # Find the retained PV
Create PVC
cat <<EOF > mariadb-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: mariadb
namespace: mariadb
spec:
accessModes:
- ReadWriteOnce
resources:
requests:

storage: 250Mi
volumeName: mariadb-pv
EOF
kubectl apply -f mariadb-pvc.yaml
Edit Deployment
nano ~/mariadb-deployment.yaml
Add volumeMount and volume to use the PVC as described
kubectl apply -f ~/mariadb-deployment.yaml
kubectl get pods -n mariadb

NEW QUESTION: 58
Scale the deployment webserver to 6 pods.
Answer:

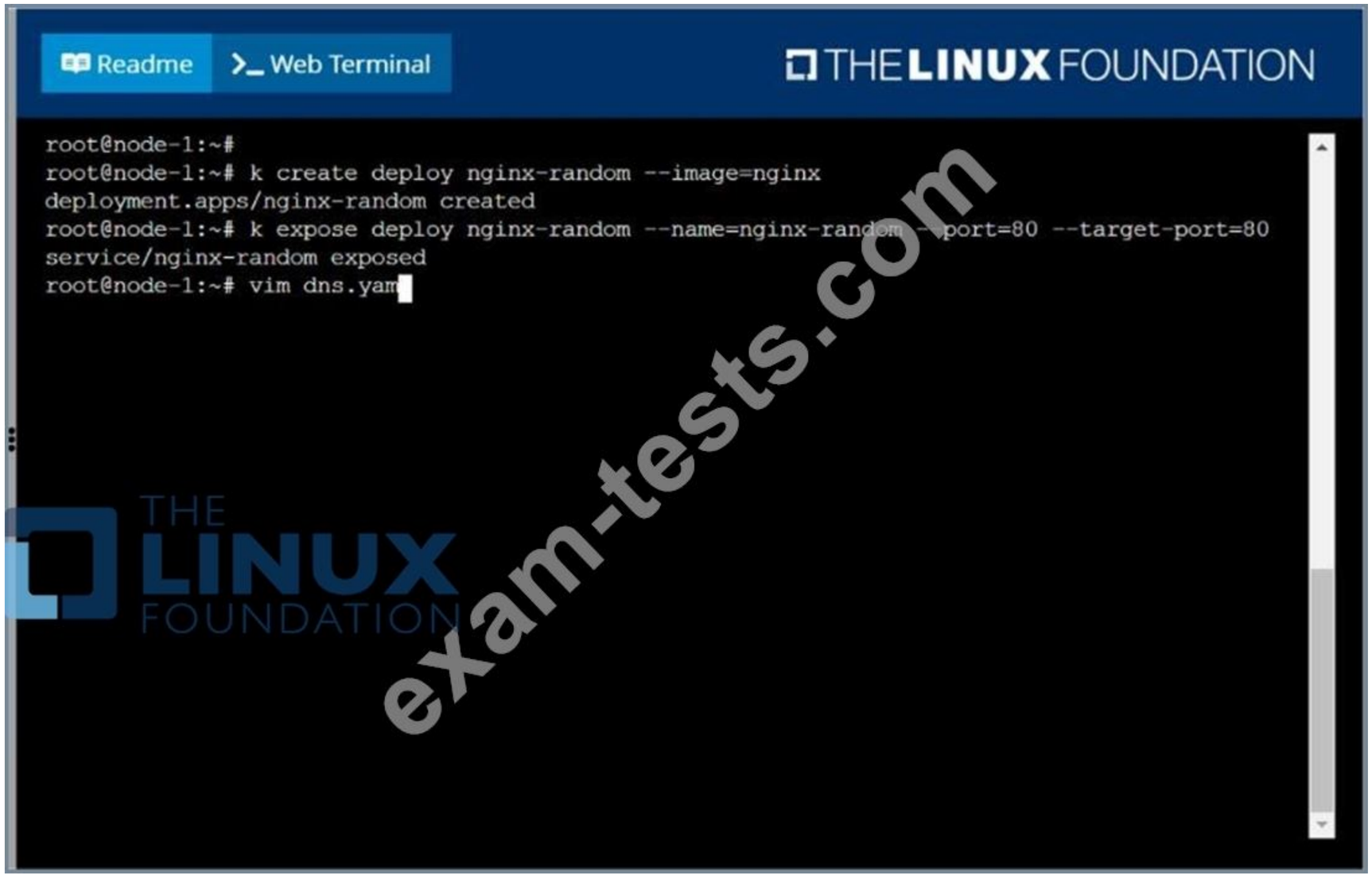


NEW QUESTION: 59
Apply the autoscaling to this deployment with minimum 10 and maximum 20 replicas and target CPU of 85% and verify hpa is created and replicas are increased to 10 from 1
Answer:

kubectl autoscale deploy webapp --min=10 --max=20 --cpu percent=85 kubectl get hpa kubectl get pod -l app=webapp

NEW QUESTION: 60
Create an nginx pod and list the pod with different levels of verbosity
Answer:
See the solution below.
Explanation
// create a pod
kubectl run nginx --image=nginx --restart=Never --port=80
// List the pod with different verbosity
kubectl get po nginx --v=7
kubectl get po nginx --v=8
kubectl get po nginx --v=9

NEW QUESTION: 61
Create a deployment as follows:
Name: nginx-random
Exposed via a service nginx-random
Ensure that the service & pod are accessible via their respective DNS records The container(s) within any pod(s) running as a part of this deployment should use the nginx Image Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to
/opt/KUNW00601/service.dns and /opt/KUNW00601/pod.dns respectively.
Answer:
See the solution below.
Explanation
Solution:
F:\Work\Data Entry Work\Data Entry\20200827\CKA\17 C.JPG



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

Readme | Web Terminal

THE LINUX FOUNDATION

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


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

NEW QUESTION: 62
Create a configmap called myconfigmap with literal value
appname=myapp
A. kubectl create cm myconfigmap --from-literal=appname=myapp
// Verify
(or)
kubectl describe cm

B. kubectl create cm myconfigmap --from-literal=appname=myapp
// Verify
kubectl get cm -o yaml
(or)
kubectl describe cm
Answer: (SHOW ANSWER)

NEW QUESTION: 63
Create a pod that having 3 containers in it? (Multi-Container)
A. image=nginx, image=redis, image=consul
Name nginx container as "nginx-container"
Name redis container as "redis-container"
Name consul container as "consul-container"
Create a pod manifest file for a container and append container
section for rest of the images
kubectl run multi-container --generator=run-pod/v1 --image=nginx --
dry-run -o yaml > multi-container.yaml
then
vim multi-container.yaml
labels:
run: multi-container
name: multi-container
spec:
containers:
- image: nginx
name: nginx-container
- image: redis
name: consul-container
restartPolicy: Always
B. image=nginx, image=redis, image=consul
Name nginx container as "nginx-container"
Name redis container as "redis-container"
Name consul container as "consul-container"
Create a pod manifest file for a container and append container
section for rest of the images
kubectl run multi-container --generator=run-pod/v1 --image=nginx --
dry-run -o yaml > multi-container.yaml
then
vim multi-container.yaml
apiVersion: v1
kind: Pod

metadata:
labels:
run: multi-container
name: multi-container
spec:
containers:
- image: nginx
name: nginx-container
- image: redis
name: redis-container
- image: consul
name: consul-container
restartPolicy: Always
Answer: (SHOW ANSWER)

NEW QUESTION: 64
Get IP address of the pod - "nginx-dev"
Answer:
See the solution below.
Explanation
Kubect1 get po -o wide
Using JsonPath
kubect1 get pods -o=jsonpath='{range
items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

NEW QUESTION: 65
Your team is deploying a critical application on Kubernetes and needs to ensure its availability and performance. You are considering implementing a load balancer for the application to distribute traffic across multiple pods. Describe the types of load balancers available in Kubernetes and explain how to implement an external load balancer using a cloud provider's load balancer service.
Answer:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1 . Types of Load Balancers in Kubernetes:
- NodePort: A simple load balancer that exposes the service on each node's IP address and a specific port.
- LoadBalancer: Exposes the service on the public IP address of the cloud provider's load balancer.
- Ingress: A higher-level abstraction that allows for more flexible routing and configuration of traffic to services.
2. Implementing an External Load Balancer using a Cloud Provider:
- Create a Kubernetes Service:
- Define a Kubernetes Service that exposes the application on a specific port.
- Configure the service type to 'LoadBalancer'.

```
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
  namespace: myapp-namespace
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: myapp
```

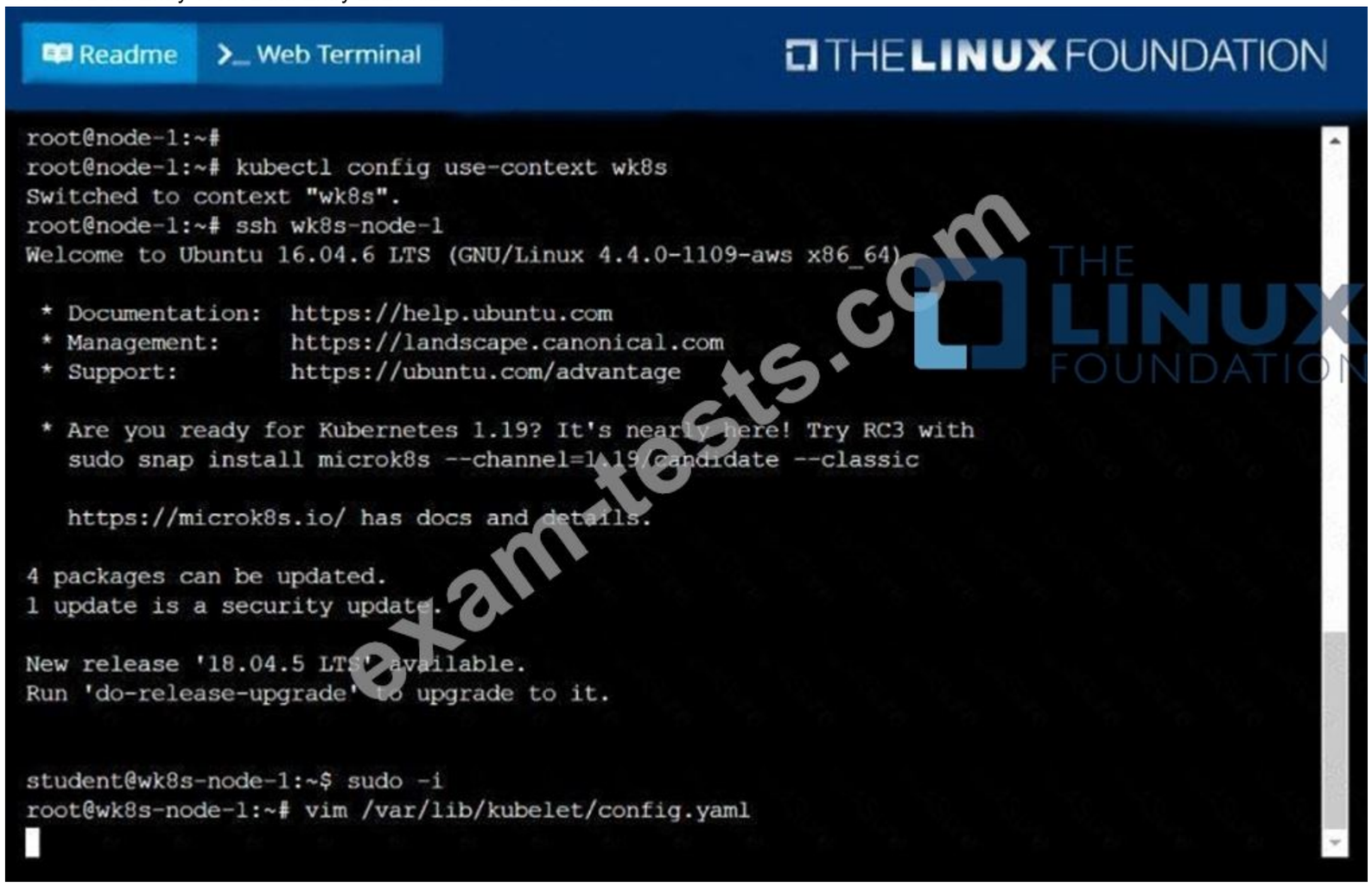

- Configure a Cloud Provider Load Balancer: - Access the load balancer management console of your cloud provider (e.g., AWS Elastic Load Balancer, Google Cloud Load Balancing, Azure Load Balancer). - Create a new load balancer and configure it to listen on the desired port (e.g., port 80). - Configure the load balancer to distribute traffic to the Kubernetes service. This might involve specifying the Kubernetes service's IP address or hostname, depending on the cloud provider's setup. - Configure the health check settings to ensure that the load balancer only routes traffic to healthy pods. - Verify Load Balancer Configuration: - Once the cloud provider load balancer is configured, verify that it is working correctly by accessing the load balancer's public IP address and ensuring that the application responds as expected. - You can also use 'kubectl describe service myapp-service' to check the load balancer's status and external IP address. ,

NEW QUESTION: 66
List pod logs named "frontend" and search for the pattern "started" and write it to a file "/opt/error-logs" See the solution below.
Answer:
Kubectl logs frontend | grep -i "started" > /opt/error-logs

NEW QUESTION: 67
What is the maximum number of samples that can be submitted to WildFire manually per day?
A. 5,000
B. 15,000
C. 1,000
D. 2,000
Answer: C (LEAVE A REPLY)

NEW QUESTION: 68
Configure the kubelet systemd- managed service, on the node labelled with name=wk8s-node-1, to launch a pod containing a single container of Image httpd named webtool automatically. Any spec files required should be placed in the /etc/kubernetes/manifests directory on the node.
You can ssh to the appropriate node using:
[student@node-1] $ ssh wk8s-node-1
You can assume elevated privileges on the node with the following command:
[student@wk8s-node-1] $ | sudo -i
Answer:
See the solution below.
Explanation

solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\21 C.JPG



F:\Work\Data Entry Work\Data Entry\20200827\CKA\21 D.JPG

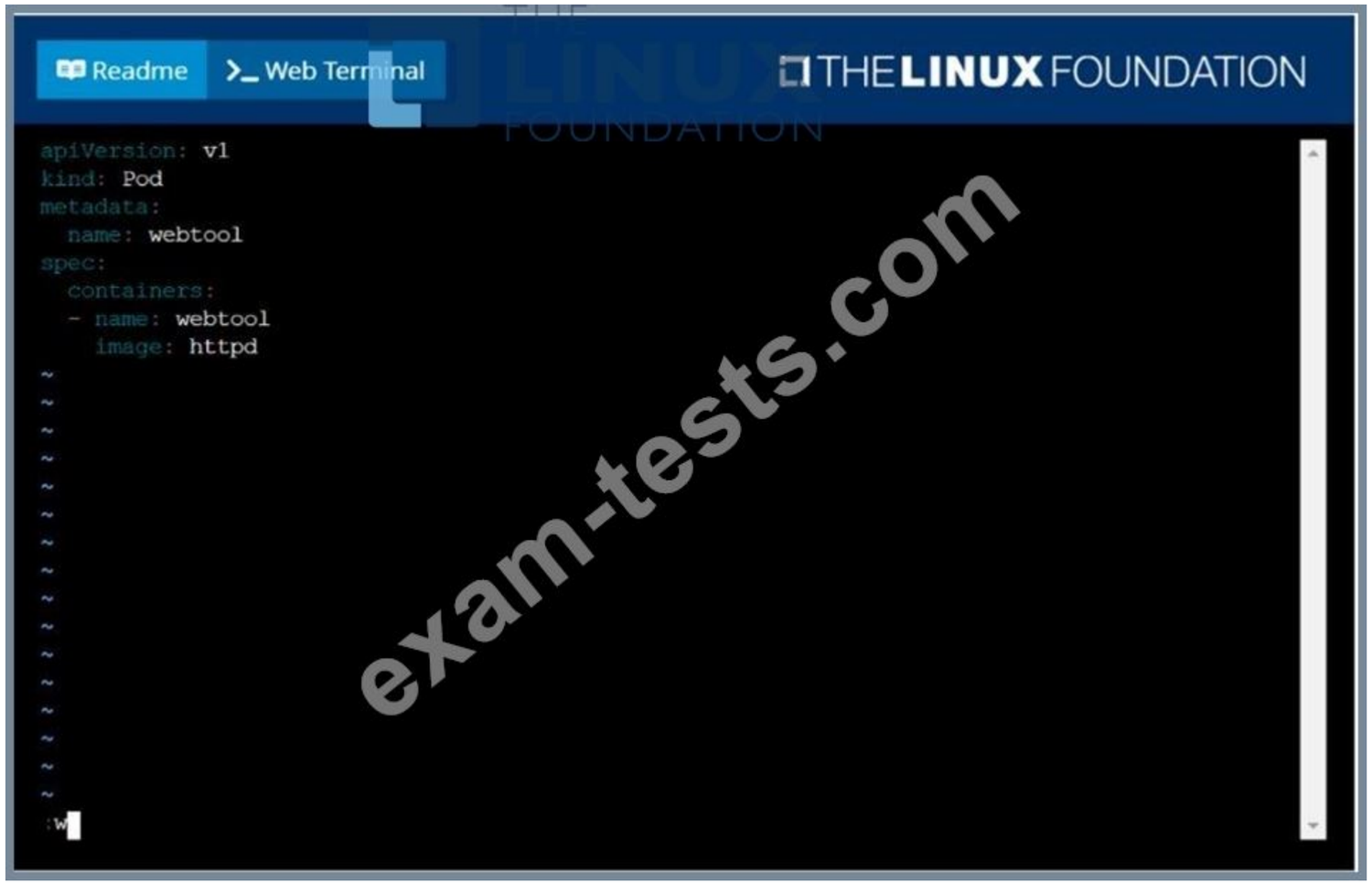
F:\Work\Data Entry Work\Data Entry\20200827\CKA\21 E.JPG

Readme | Web Terminal

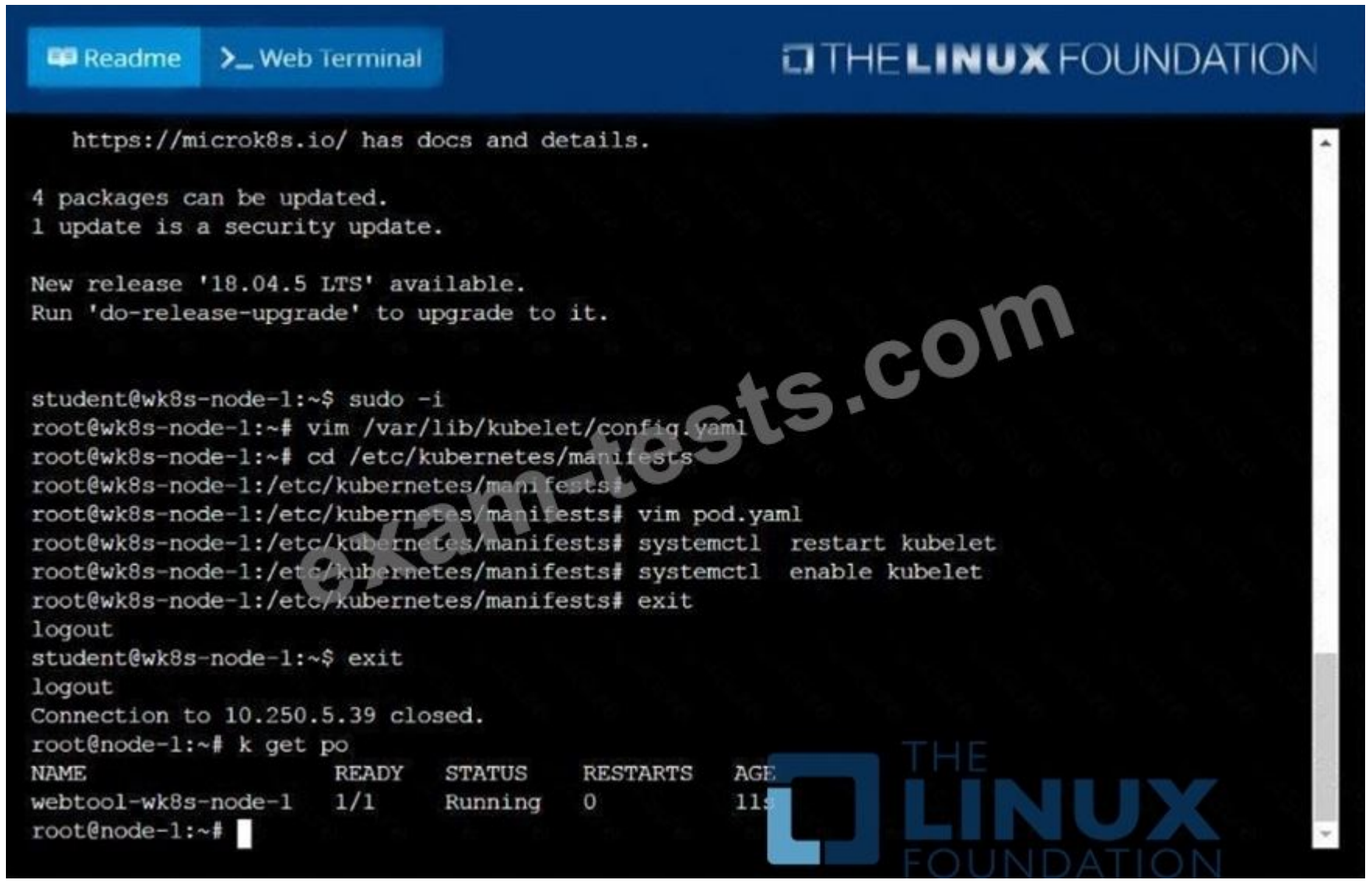
THE LINUX FOUNDATION

```
root@node-1:~# ssh wk8s-node-1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1109-aws x86_64)

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

 * Are you ready for Kubernetes 1.19? It's nearly here! Try RC3 with
   sudo snap install microk8s --channel=1.19/candidate --classic

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

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

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


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

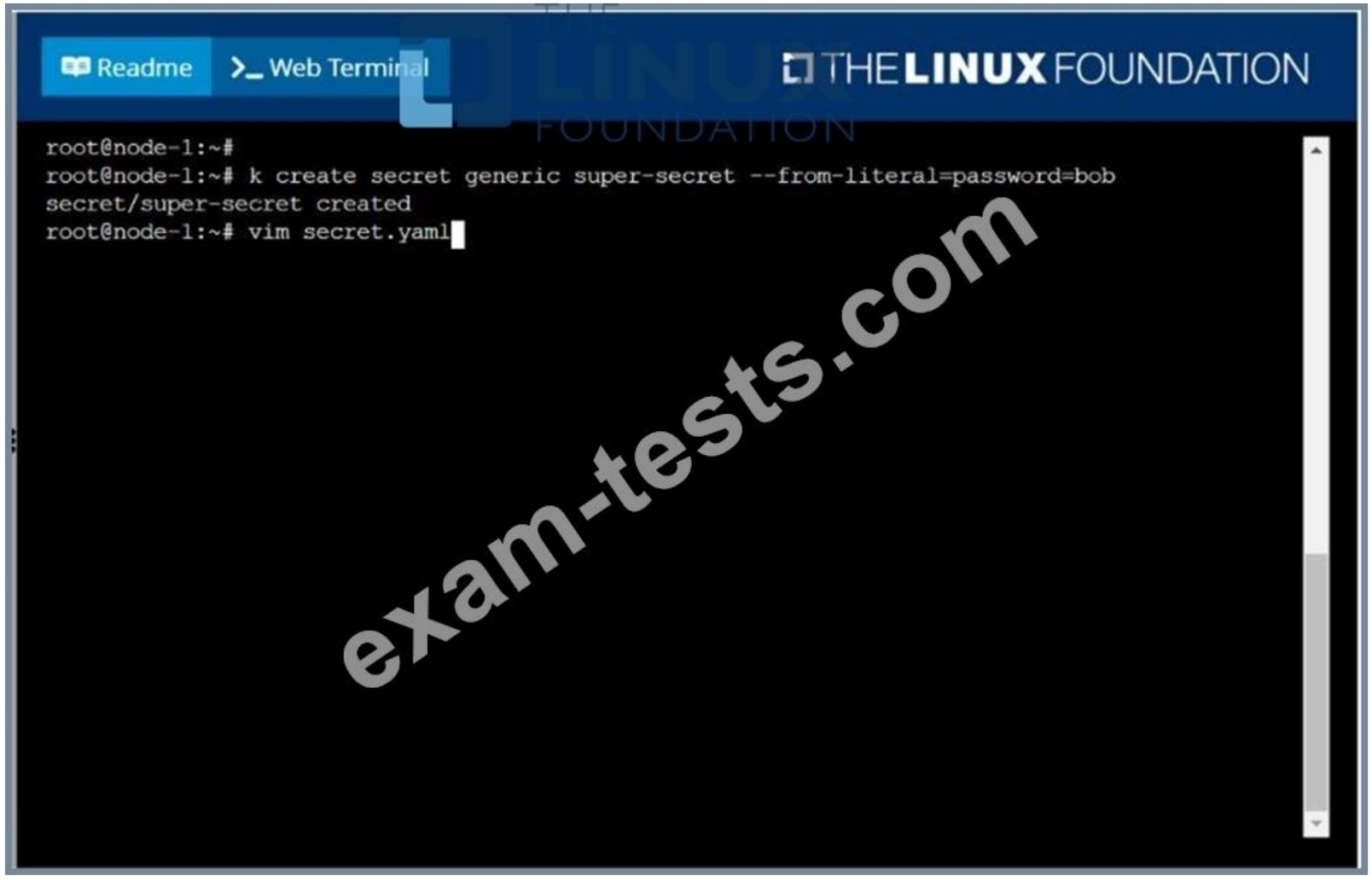

F:\Work\Data Entry Work\Data Entry\20200827\CKA\21 F.JPG



F:\Work\Data Entry Work\Data Entry\20200827\CKA\21 G.JPG



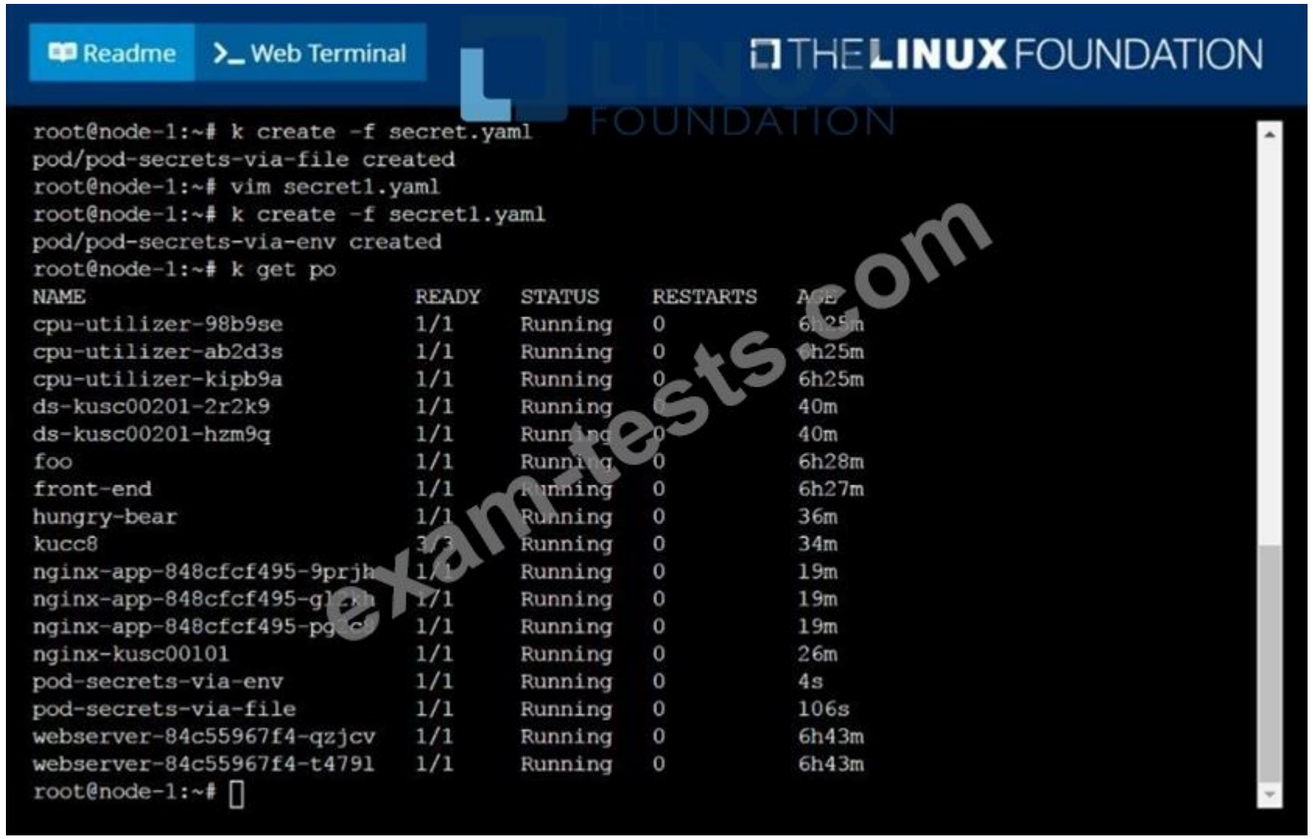
NEW QUESTION: 69
Create a Kubernetes secret as follows:
* Name: super-secret
* password: bob
Create a pod named pod-secrets-via-file, using the redis Image, which mounts a secret named super-secret at /secrets.
Create a second pod named pod-secrets-via-env, using the redis Image, which exports password as CONFIDENTIAL.
Answer:


Readme
Web Terminal
THE LINUX FOUNDATION
root@node-1:~#
root@node-1:~# k create secret generic super-secret --from-literal=password=bob
secret/super-secret created
root@node-1:~# vim secret.yaml
exam-tests.com

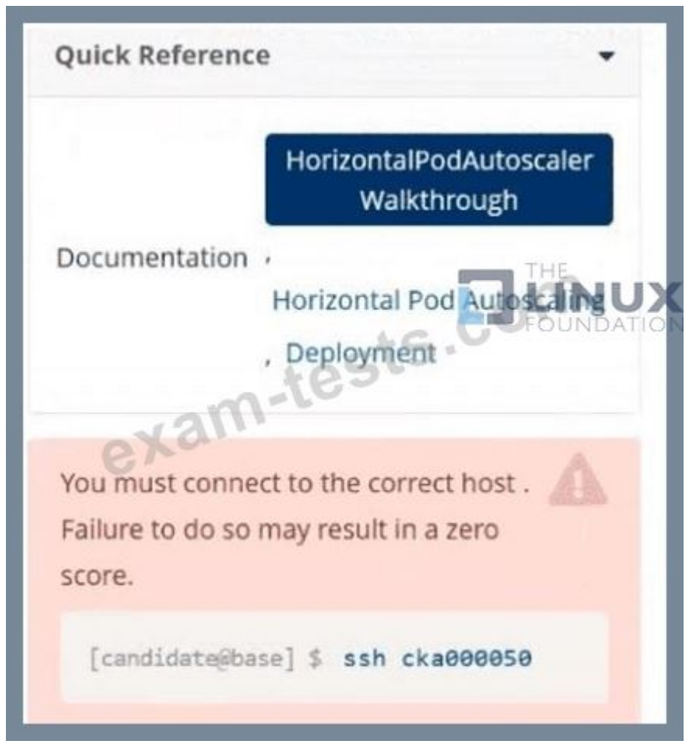
Readme | Web Terminal

THE LINUX FOUNDATION

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



NEW QUESTION: 70



Task

Create a new HorizontalPodAutoscaler (HPA) named apache-server in the autoscale namespace. This HPA must target the existing Deployment called apache-server in the autoscale namespace.

Set the HPA to aim for 50% CPU usage per Pod . Configure it to have at least 1 Pod and no more than 4 Pods

. Also, set the downscale stabilization window to 30 seconds.

Answer:

Task Summary

* Create an HPA named apache-server in the autoscale namespace.
* Target an existing deployment also named apache-server.
* CPU target: 50%
* Pod range: min 1, max 4
* Downscale stabilization window: 30 seconds

Step-by-Step Answer

Step 1: Connect to the correct host

This is critical, as shown in the warning image.

ssh cka000050

Skipping this may result in zero for this question!

Step 2: Verify the deployment exists

kubectl get deployment apache-server -n autoscale

Make sure it's there before creating the HPA. If it's missing, the HPA won't bind correctly.

Step 3: Create the HPA

We will use the kubectl autoscale command for a quick setup, then patch it to add the stabilization window (since kubectl autoscale doesn't include it).

kubectl autoscale deployment apache-server \
--namespace autoscale \
--cpu-percent=50 \
--min=1 \
--max=4
Step 4: Add the downscale stabilization window
You'll need to patch the HPA to include the stabilization window of 30s.
Create a patch file called hpa-patch.yaml:
spec:
behavior:
scaleDown:
stabilizationWindowSeconds: 30
Apply the patch:
bash
CopyEdit
kubectl patch hpa apache-server \
-n autoscale \
--patch "$(cat hpa-patch.yaml)"
Step 5: Confirm your work
bash
CopyEdit
kubectl describe hpa apache-server -n autoscale
Look for:
* Min/Max Pods: 1/4
* Target CPU utilization: 50%
* Stabilization window: should appear under Behavior > ScaleDown
ssh cka000050
kubectl get deployment apache-server -n autoscale
kubectl autoscale deployment apache-server \
--namespace autoscale \
--cpu-percent=50 \
--min=1 \
--max=4
Patch to add stabilization window
cat <<EOF > hpa-patch.yaml
spec:
behavior:
scaleDown:
stabilizationWindowSeconds: 30
EOF
kubectl patch hpa apache-server -n autoscale --patch "$(cat hpa-patch.yaml)"

NEW QUESTION: 71
Undo the deployment to the previous version 1.17.1 and verify Image has the previous version
Answer:
kubectl rollout undo deploy webapp kubectl describe deploy webapp | grep Image

NEW QUESTION: 72
Add a taint to node "worker-2" with effect as "NoSchedule" and
list the node with taint effect as "NoSchedule"
A. // Add taint to node "worker-2"
kubectl taint nodes worker-2 key=value:NoSchedule
// Verify
// Using "custom-coloumns" , you can customize which coloumn to
be printed
kubectl get nodes -o customcolumns=NAME:.metadata.name,TAINTS:.spec.taints --no-headers
// Using jsonpath
kubectl get nodes -o jsonpath="{range
.items[*]}{.metadata.name} {.spec.taints[?(
@.effect=='NoSchedule')].effect}{\"\n\"}{end}" | awk 'NF==2
{print $0}'
B. // Add taint to node "worker-2"
kubectl taint nodes worker-2 key=value:NoSchedule
.items[*]}{.metadata.name} {.spec.taints[?(
@.effect=='NoSchedule')].effect}{\"\n\"}{end}" | awk 'NF==2
{print $0}'
Answer: A (LEAVE A REPLY)

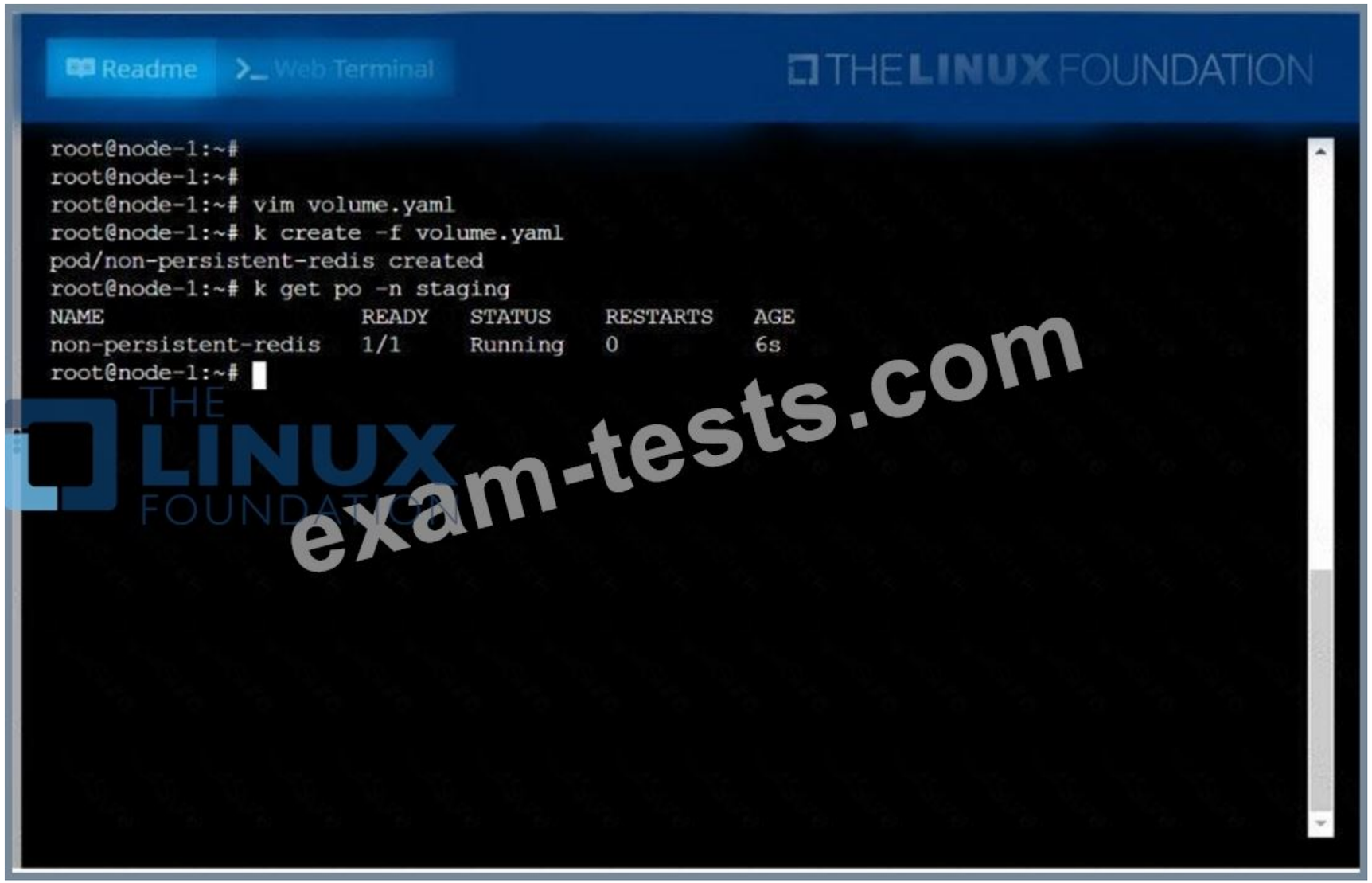
NEW QUESTION: 73
Create a pod as follows:
* Name:non-persistent-redis
* container Image:redis
* Volume with name:cache-control
* Mount path:/data/redis
The pod should launch in thestagingnamespace and the volumemust notbe persistent.
Answer:
See the solution below.
Explanation
solution


Readme
>_ Web Terminal
THE LINUX FOUNDATION
root@node-1:~#
root@node-1:~#
root@node-1:~# vim volume.yaml
exam-tests.com
THE
LINUX
FOUNDATION

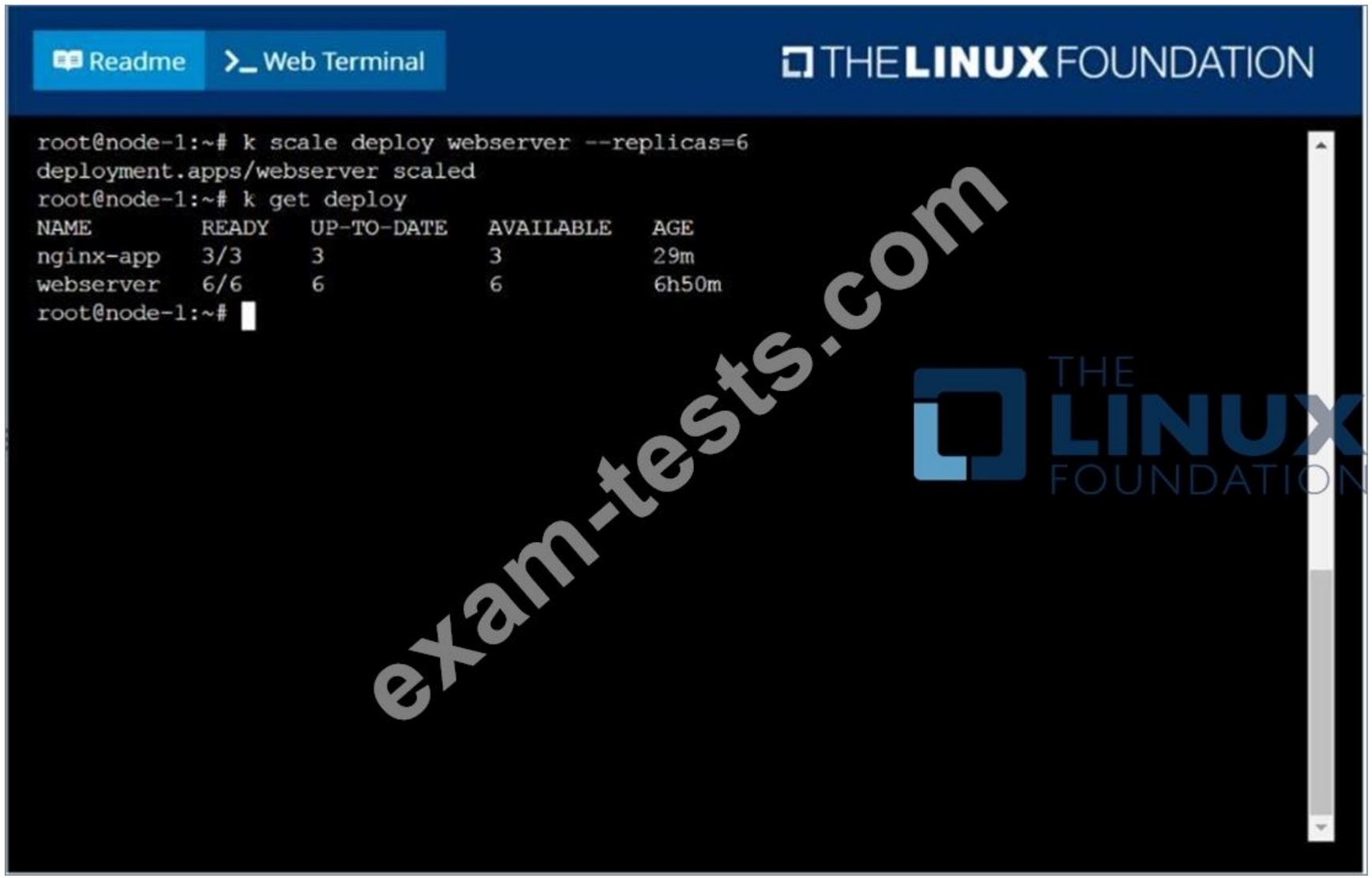
Readme | Web Terminal

THE LINUX FOUNDATION

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



NEW QUESTION: 74
Scale the deployment webserver to
Answer:
See the solution below.
Explanation
solution
F:\Work\Data Entry Work\Data Entry\20200827\CKA\14 B.JPG



NEW QUESTION: 75
Create a deployment called webapp with image nginx having 5 replicas in it, put the file in /tmp directory with named webapp.yaml
A. //Create a file using dry run command
kubectl create deploy --image=nginx --dry-run -o yaml >
/tmp/webapp.yaml
// Now, edit file webapp.yaml and update replicas=5
apiVersion: apps/v1
kind: Deployment
metadata:
labels:

app: webapp
name: webapp
spec:
replicas: 5
selector:
matchLabels:
app: webapp
template:
metadata:
labels:
app: webapp
spec:
containers:
- image: nginx
name: nginx
Note: Search "deployment" in kubernetes.io site , you will get
the page
https://kubernetes.io/docs/concepts/workloads/controllers/deplo
yment/
// Verify the Deployment
kubectl get deploy webapp --show-labels
// Output the YAML file of the deployment webapp
kubectl get deploy webapp -o yaml
B. //Create a file using dry run command
kubectl create deploy --image=nginx --dry-run -o yaml >
/tmp/webapp.yaml
// Now, edit file webapp.yaml and update replicas=5
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: webapp
name: webapp
spec:
replicas: 5
selector:
matchLabels:
app: webapp
template:
metadata:
labels:

Note: Search "deployment" in kubernetes.io site , you will get
the page
https://kubernetes.io/docs/concepts/workloads/controllers/deplo
yment/
// Verify the Deployment
kubectl get deploy webapp --show-labels
// Output the YAML file of the deployment webapp
kubectl get deploy webapp -o yaml
Answer: A (LEAVE A REPLY)

NEW QUESTION: 76
You need to create a new role that allows users to create and delete pods, but only in the 'production' namespace. How would you define this role using the 'kubectl' command?
Answer:
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Role YAML file:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-admin
  namespace: production
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["deployments/finalizers"]
  verbs: ["update"]
- apiGroups: ["apps"]
  resources: ["statefulsets"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["statefulsets/finalizers"]
  verbs: ["update"]
- apiGroups: ["extensions"]
  resources: ["daemonsets"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["daemonsets/finalizers"]
  verbs: ["update"]
- apiGroups: ["extensions"]
  resources: ["ingresses"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["ingresses/finalizers"]
  verbs: ["update"]
- apiGroups: ["extensions"]
  resources: ["replicasets"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["replicasets/finalizers"]
  verbs: ["update"]
- apiGroups: ["batch"]
  resources: ["jobs"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch"]
  resources: ["jobs/finalizers"]
  verbs: ["update"]
- apiGroups: ["batch"]
  resources: ["cronjobs"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch"]
  resources: ["cronjobs/finalizers"]
  verbs: ["update"]
- apiGroups: ["", "apps", "extensions", "batch"]
  resources: ["pods"]
  verbs: ["get", "list", "watch", "create", "delete", "update"]
- apiGroups: ["", "apps", "extensions", "batch"]
  resources: ["pods/status"]
  verbs: ["get"]
- apiGroups: ["apps"]
  resources: ["deployments/status"]
  verbs: ["get"]
- apiGroups: ["apps"]
  resources: ["statefulsets/status"]
```

```
  resources: ["statefulsets/status"]
  verbs: ["get"]
- apiGroups: ["extensions"]
  resources: ["daemonsets/status"]
  verbs: ["get"]
- apiGroups: ["extensions"]
  resources: ["replicasets/status"]
  verbs: ["get"]
- apiGroups: ["batch"]
  resources: ["jobs/status"]
  verbs: ["get"]
- apiGroups: ["batch"]
  resources: ["cronjobs/status"]
  verbs: ["get"]
- apiGroups: ["extensions"]
  resources: ["ingresses/status"]
  verbs: ["get"]
```


2. Apply the Role to the cluster: kubectl apply -f pod-admin . yaml 3. Create a RoleBinding to associate the Role with a user or group:

```
kubectl create rolebinding pod-admin-binding --role=pod-admin --subjects=user:user1--namespace=production
```

The 'Role' resource defines a set of permissions for a specific namespace. The 'rules' field defines the actions allowed for the role, specifying the API groups, resources, and verbs. The 'apiGroups' field lists the Kubernetes API groups relevant to the permissions. The 'resources' field specifies the Kubernetes resources that the user can access. The 'verbs' field lists the allowed actions for the specified resources. The 'RoleBinding' associates the created 'Role' with a specific user or group, granting them the specified permissions. In this case, the role is named "pod-admin" and is scoped to the 'production' namespace. The role allows users to create and delete pods, as well as perform other actions on related resources. This example demonstrates how to manage Role-Based Access Control (RBAC) in Kubernetes. You can adjust the permissions and bindings to fit your specific security requirements.,

NEW QUESTION: 77
Create a deployment as follows:
* Name: nginx-random
* Exposed via a service nginx-random
* Ensure that the service & pod are accessible via their respective DNS records
* The container(s) within any pod(s) running as a part of this deployment should use the nginx Image Next, use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/KUNW00601/service.dns and /opt/KUNW00601/pod.dns respectively.
Answer:
See the solution below.
Explanation
Solution:

Readme | Web Terminal

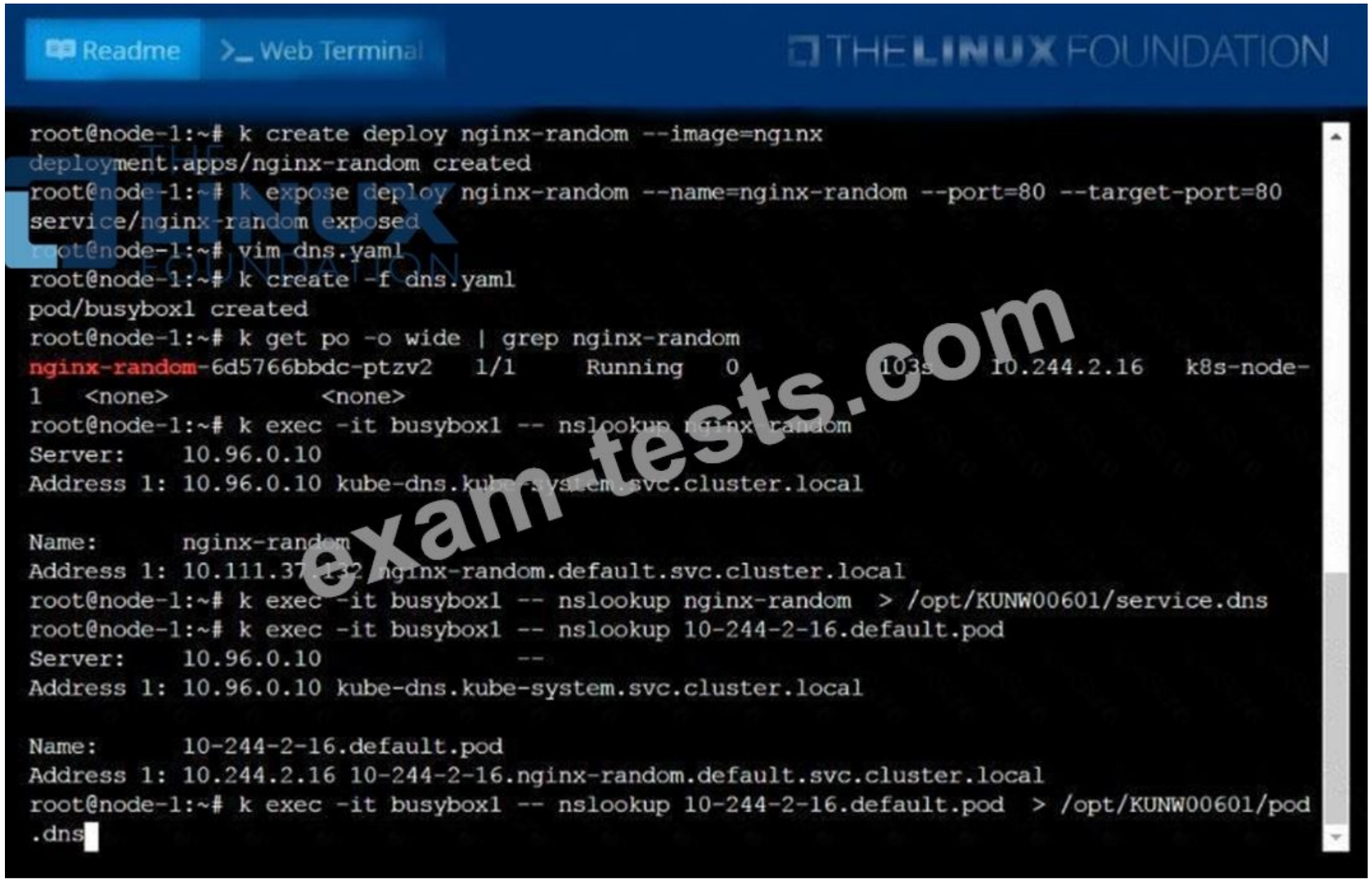
THE LINUX FOUNDATION

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

Readme | >_ Web Terminal

THE LINUX FOUNDATION

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



NEW QUESTION: 78
To protect your firewall and network from single source denial of service (DoS) attacks that can overwhelm its packet buffer and cause legitimate traffic to drop, you can configure:
A. PBP (Protocol Based Protection)
B. PGP (Packet Gateway Protocol)
C. PBP (Packet Buffer Protection)
D. BGP (Border Gateway Protocol)
Answer: (SHOW ANSWER)

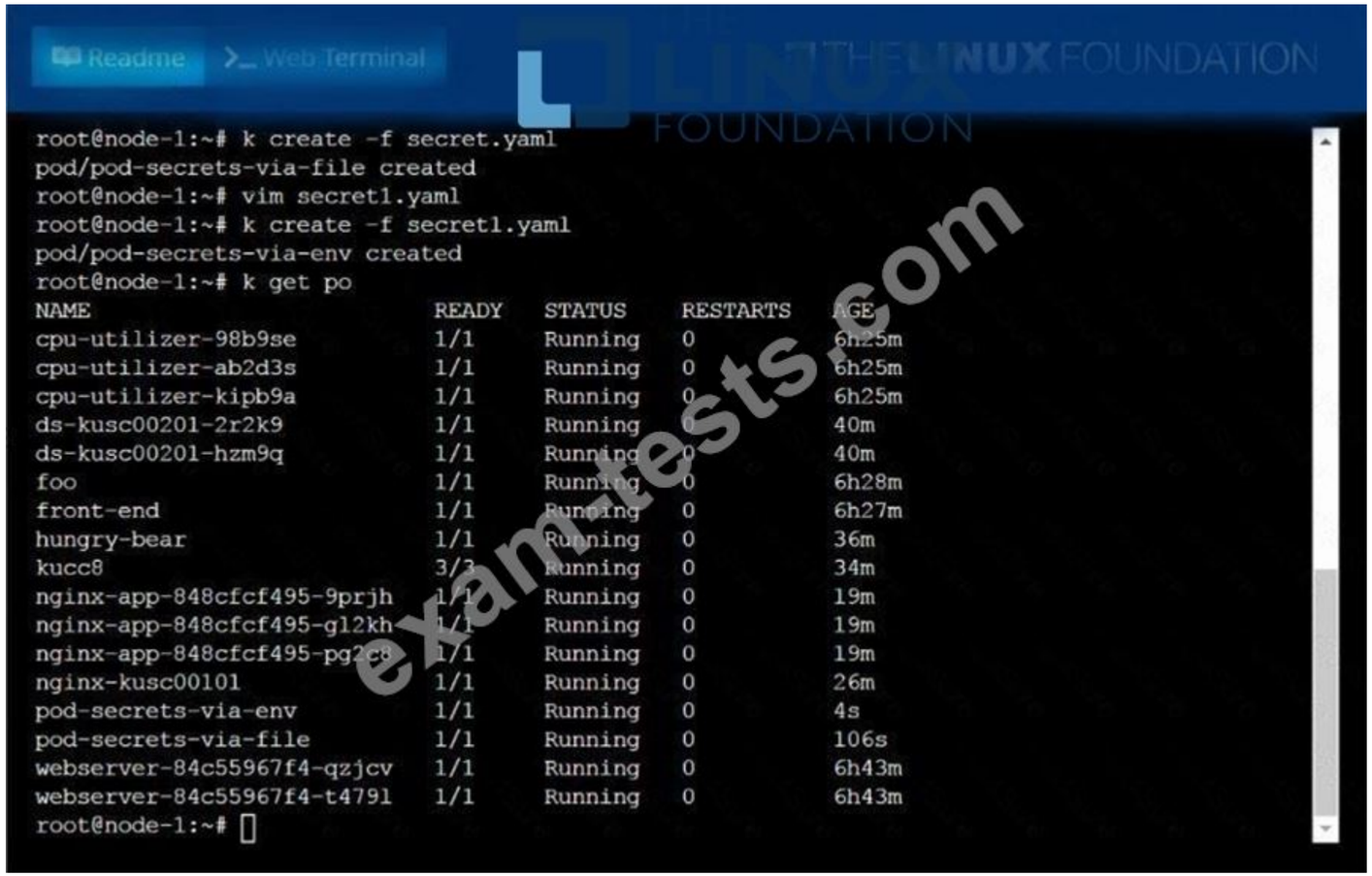
NEW QUESTION: 79
Create a Kubernetes secret asfollows:
* Name: super-secret

* password: bob
Create a pod namedpod-secrets-via-file Image, which mounts a secret namedsuper-secretat
/secrets.
Create a second pod namedpod-secrets-via-env Image, which exportspasswordas CONFIDENTIAL
Answer:
See the solution below.
Explanation
solution

Readme | >_ Web Terminal

THE LINUX FOUNDATION

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

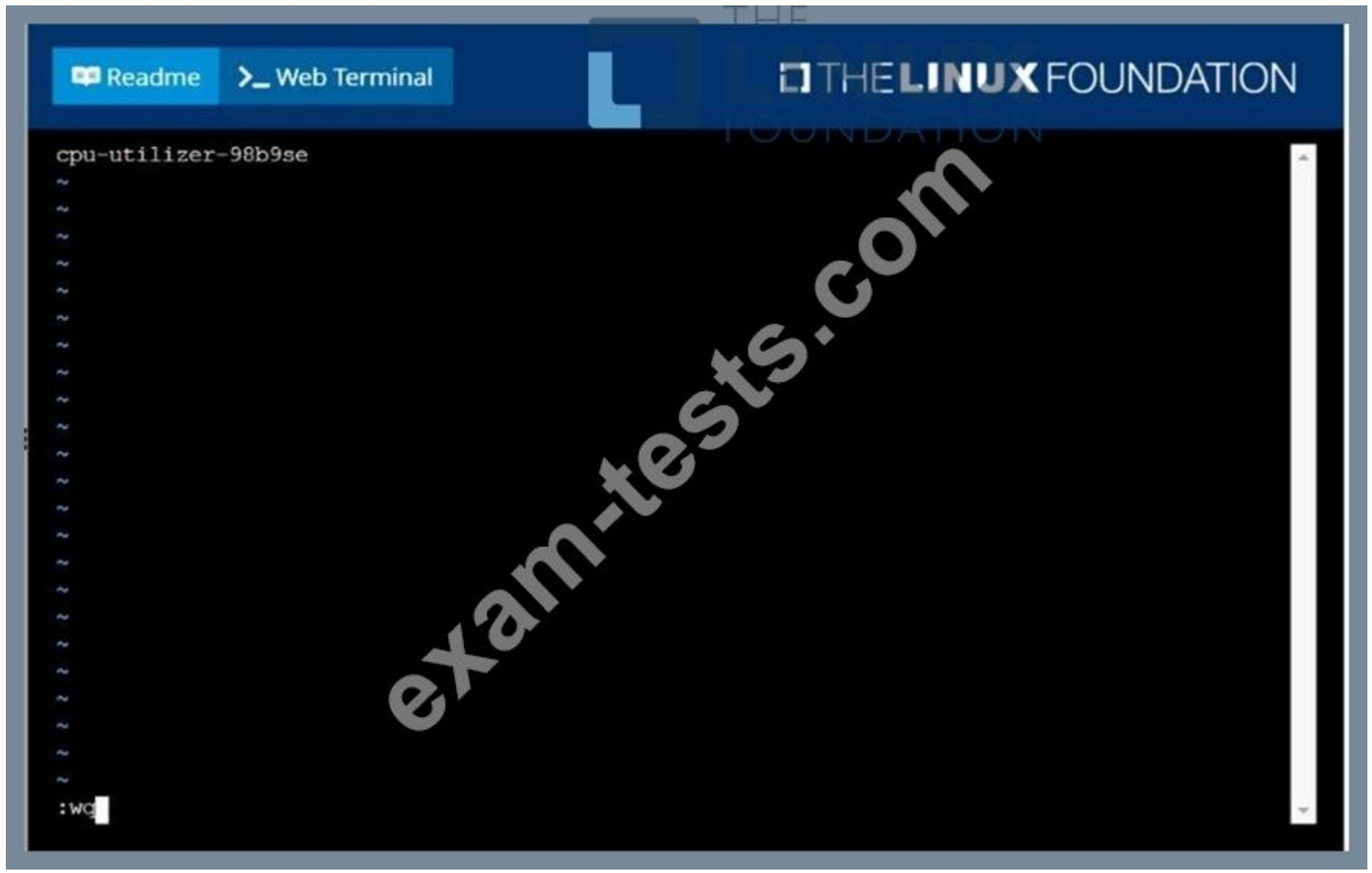


NEW QUESTION: 80
Create an nginx pod which reads username as the environment variable
A. // create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx.yml
// add env section below and create
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx

env:
- name: USER_NAME
valueFrom:
secretKeyRef:
name: my-secret
key: username
restartPolicy: Never
kubectl create -f nginx.yml
//Verify
kubectl exec -it nginx - env
B. // create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx.yml
// add env section below and create
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
env:
- name: USER_NAME
restartPolicy: Never
kubectl create -f nginx.yml
//Verify
kubectl exec -it nginx - env
Answer: A (LEAVE A REPLY)

NEW QUESTION: 81
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/KUTR00102/KUTR00102.txt (which already exists).
Answer:
solution


Readme
Web Terminal
THE LINUX FOUNDATION
root@node-1:~# k top po -l name=cpu-utilizer
NAME CPU(cores) MEMORY(bytes)
cpu-utilizer-98b9se 60m 7Mi
cpu-utilizer-ab2d3s 14m 7Mi
cpu-utilizer-kipb9a 45m 7Mi
root@node-1:~# vim /opt/KUTR00102/KUTR00102.txt
THE LINUX FOUNDATION
exam-tests.com



NEW QUESTION: 82
Create a daemonset named "Prometheus-monitoring" using image=prom/Prometheus which runs in all the nodes in the cluster. Verify the pod running in all the nodes
A. vim promo-ds.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: prometheus-monitoring
spec:
selector:
matchLabels:
name: prometheus
template:
metadata:
labels:
name: prometheus
spec:
tolerations:

remove it if your masters can't run pods
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: prometheus-container
image: prom/prometheus
volumeMounts:
- name: varlog
mountPath: /var/log
- name: varlibdockercontainers
mountPath: /var/lib/docker/containers
readOnly: true
volumes:
- name: varlog
emptyDir: {}
- name: varlibdockercontainers
emptyDir: {}
kubectl apply -f promo-ds.yaml
NOTE: Deamonset will get scheduled to "default" namespace, to
schedule deamonset in specific namespace, then add
"namespace" field in metadata
//Verify
kubectl get ds
NAME DESIRED CURRENT READY UP-TO-DATE
AVAILABLE NODE SELECTOR AGE
prometheus-monitoring 6 6 0 6
0 <none> 7s
kubectl get no # To get list of nodes in the cluster
// There are 6 nodes in the cluster, so a pod gets scheduled to
each node in the cluster
B. vim promo-ds.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
name: prometheus-monitoring
spec:
selector:
matchLabels:
name: prometheus
template:
metadata:

labels:
name: prometheus
spec:
tolerations:
remove it if your masters can't run pods
- key: node-role.kubernetes.io/master
effect: NoSchedule
containers:
- name: prometheus-container
- name: varlibdockercontainers
mountPath: /var/lib/docker/containers
readOnly: true
volumes:
- name: varlog
emptyDir: {}
- name: varlibdockercontainers
emptyDir: {}
kubectl apply -f promo-ds.yaml
NOTE: Deamonset will get scheduled to "default" namespace, to
schedule deamonset in specific namespace, then add
"namespace" field in metadata
//Verify
kubectl get ds
NAME DESIRED CURRENT READY UP-TO-DATE
AVAILABLE NODE SELECTOR AGE
prometheus-monitoring 8 8 0 6
0 <none> 7s
kubectl get no # To get list of nodes in the cluster
// There are 6 nodes in the cluster, so a pod gets scheduled to
each node in the cluster
Answer: A (LEAVE A REPLY)

NEW QUESTION: 83
You must connect to the correct host.
Failure to do so may result in a zero score.
[candidate@base] $ ssh Cka000054
Context:
Your cluster 's CNI has failed a security audit. It has been removed. You must install a new CNI that can enforce network policies.
Task
Install and set up a Container Network Interface (CNI) that meets these requirements:
Pick and install one of these CNI options:

Flannel version 0.26.1
Manifest:
https://github.com/flannel-io/flannel/releases/download/v0.26.1/kube-flannel.yml
Calico version 3.28.2
Manifest:
https://raw.githubusercontent.com/project calico/calico/v3.28.2/manifests/tigera-operator.yaml
Answer:
Task Summary
* SSH into cka000054
* Install a CNI plugin that supports NetworkPolicies
* Two CNI options provided:
* Flannel v0.26.1 (# does NOT support NetworkPolicies)
* Calico v3.28.2 # (does support NetworkPolicies)
# Decision Point: Which CNI to choose?
# Choose Calico, because only Calico supports enforcing NetworkPolicies natively. Flannel does not.
# Step-by-Step Solution
1## SSH into the correct node
ssh cka000054
Required. Skipping this results in zero score.
2## Install Calico CNI (v3.28.2)
Use the official manifest provided:
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/tigera-operator.yaml This installs the Calico Operator, which then deploys the full Calico CNI stack.
3## Wait for Calico components to come up
Check the pods in tigera-operator and calico-system namespaces:
kubectl get pods -n tigera-operator
kubectl get pods -n calico-system
You should see pods like:
* calico-kube-controllers
* calico-node
* calico-typha
* tigera-operator
Wait for all to be in Running state.
(Optional) 4## Confirm CNI is enforcing NetworkPolicies
You can check:
kubectl get crds | grep networkpolicy
You should see:
* networkpolicies.crd.projectcalico.org
* This confirms Calico's CRDs are installed for policy enforcement.
Final Command Summary
ssh cka000054
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/tigera-operator.yaml kubectl get pods -n tigera-operator kubectl get pods -n calico-system kubectl get crds | grep

networkpolicy

NEW QUESTION: 84
Which one is the correct configuration?
A. #Panorama
B. $Panorama
C. &Panorama
D. @Panorama
Answer: B (LEAVE A REPLY)

NEW QUESTION: 85
Create the service as type NodePort with the port 32767 for the nginx pod with the pod selector app: my-nginx
Answer:
kubectl run nginx --image=nginx --restart=Never -- labels=app=nginx --port=80 --dry-run -o yaml > nginx-pod.yaml

NEW QUESTION: 86
Delete the pod without any delay (force delete)
Answer:
Kubect1 delete po "POD-NAME" --grace-period=0 --force

NEW QUESTION: 87
Score: 4%

Task
Set the node named ek8s-node-1 as unavailable and reschedule all the pods running on it.
Answer:
See the solution below.
Explanation
SOLUTION:
[student@node-1] > ssh ek8s
kubectl cordon ek8s-node-1
kubectl drain ek8s-node-1 --delete-local-data --ignore-daemonsets --force
NEW QUESTION: 88
Score:7%

Context

An existing Pod needs to be integrated into the Kubernetes built-in logging architecture (e. g. kubectl logs).

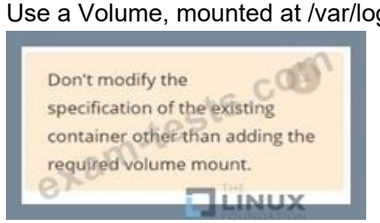
Adding a streaming sidecar container is a good and common way to accomplish this requirement.

Task

Add a sidecar container named sidecar, using the busybox Image, to the existing Pod big-corp-app. The new sidecar container has to run the following command:

/bin/sh -c tail -n+1 -f /va r/log/big-corp-app.log

Use a Volume, mounted at /var/log, to make the log file big-corp-app.log available to the sidecar container.



Answer:

Solution:

#

kubectl get pod big-corp-app -o yaml

#

apiVersion: v1

kind: Pod

metadata:

name: big-corp-app

spec:

containers:

- name: big-corp-app

image: busybox

args:

- /bin/sh

- -c

- >

i=0;

while true;

do

echo "$(date) INFO $i" >> /var/log/big-corp-app.log;

```
i=$((i+1));
sleep 1;
done
volumeMounts:
- name: logs
mountPath: /var/log
- name: count-log-1
image: busybox
args: [/bin/sh, -c, 'tail -n+1 -f /var/log/big-corp-app.log']
volumeMounts:
- name: logs
mountPath: /var/log
volumes:
- name: logs
emptyDir: {
}
#
kubectl logs big-corp-app -c count-log-1
```