

# Linux Foundation Linux Foundation CKA PDF

## Quick Reference

### HorizontalPodAutoscaler Walkthrough

## Documentation

### Horizontal Pod Autoscaling

#### 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.

#### Options:

A. 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 "$(cat hpa-patch.yaml)"` Step 5: Confirm your work `bash CopyEdit kubectl describe hpa apache-server -n autoscale` Look for: Min/Max Pods: 1/4 Target CPU utilization: 50% Stabilization window: should appear under Behavior > ScaleDown `ssh cka000050 kubectl get deployment apache-server -n autoscale kubectl autoscale deployment apache-server \ --namespace autoscale \ --cpu-percent=50 \ --min=1 \ --max=4 # Patch to add stabilization window cat hpa-patch.yaml spec: behavior: scaleDown: stabilizationWindowSeconds: 30 EOF kubectl patch hpa apache-server -n autoscale --patch "$(cat hpa-patch.yaml)"`

## Answer: A

### Explanation:

#### 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 HP

A. 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

Readme

Web Terminal

THE LINUX FOUNDATION

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

F:\Work\Data Entry Work\Data Entry\20200827\CKA\1 B.JPG

F:\Work\Data Entry Work\Data Entry\20200827\CKA\1 C.JPG

Step 0: Set the correct Kubernetes context

If you're given a specific context (k8s in this case), you must switch to it:

kubectl config use-context k8s

Skipping this can cause you to work in the wrong cluster/namespace and cost you marks.

Step 1: Identify the namespace of the pod foo

First, check if foo is running in a specific namespace or in the default namespace.

```
kubectl get pods --all-namespaces | grep foo
```

Assume the pod is in the default namespace if no namespace is mentioned.

Step 2: Confirm pod foo exists and is running

```
kubectl get pod foo
```

You should get output similar to:

```
NAME READY STATUS RESTARTS AGE
```

```
foo 1/1 Running 0 1h
```

Readme > Web Terminal

THE LINUX FOUNDATION

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

## SIMULATION

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

### Options:

A. solution

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

**Answer: A**

## Question 4

### SIMULATION

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

The screenshot shows a web terminal interface with a dark background. At the top, there is a blue header bar with the text 'THE LINUX FOUNDATION' on the right. On the left of the header, there are two buttons: 'Readme' and 'Web Terminal'. The main area of the terminal displays a YAML configuration for a Kubernetes pod. The configuration includes a 'kind' of 'Pod', a 'name' of 'hungry-bear', and a 'spec' with 'volumes' and 'containers'. The first container, named 'checker', uses the 'alpine' image and runs a command that checks for the existence of '/workdir/calm.txt'. If it exists, it sleeps for 100,000 seconds; otherwise, it exits with a status of 1. The second container, named 'create', also uses the 'alpine' image and runs a command to create the file '/workdir/calm.txt'. Both containers have a volume mount for 'workdir' at the path '/workdir'. The terminal ends with a prompt ':wg'.

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

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

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

### Options:

A. solution

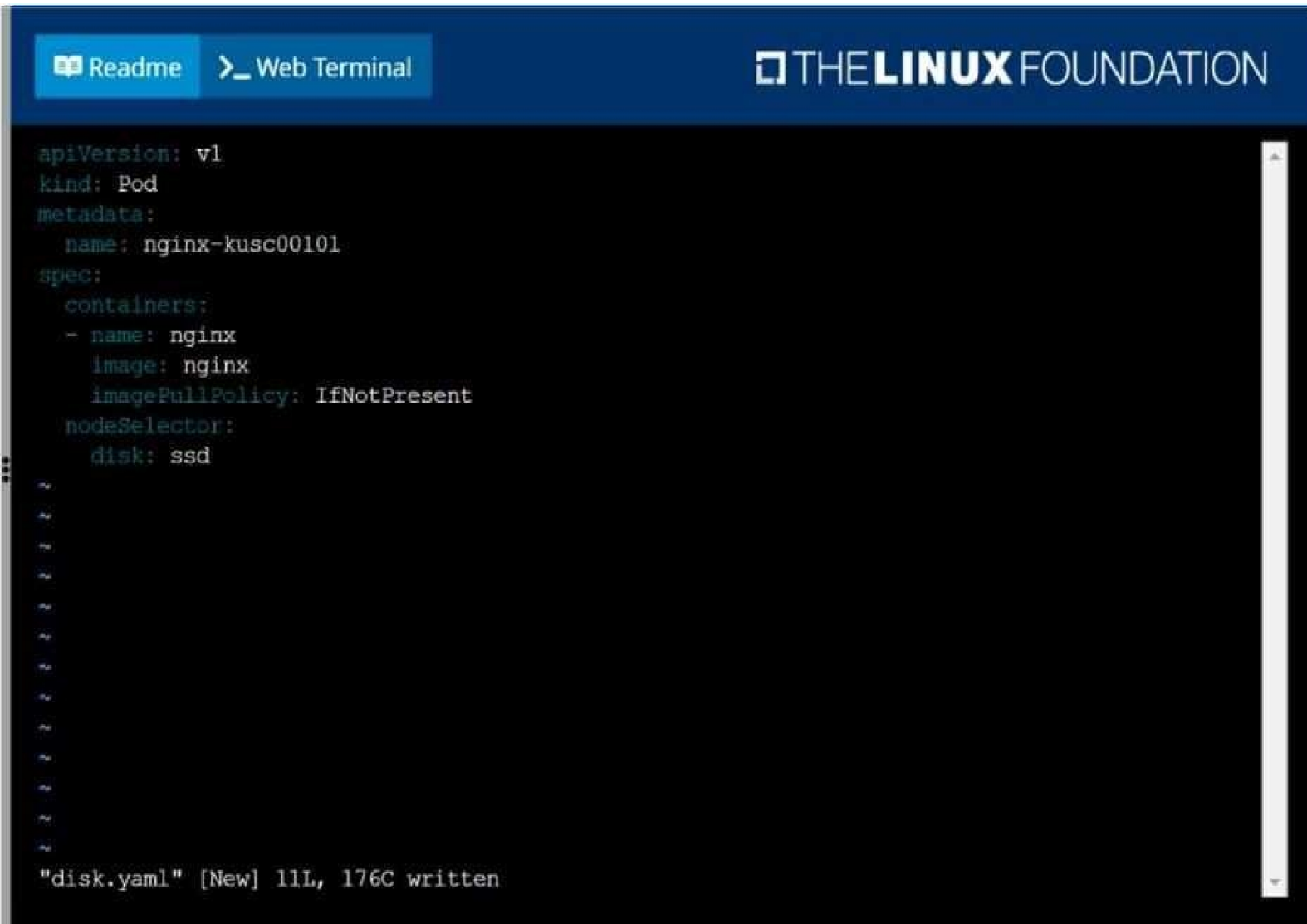
F:\Work\Data Entry Work\Data Entry\20200827\CKA\4 B.JPG

F:\Work\Data Entry Work\Data Entry\20200827\CKA\4 C.JPG

F:\Work\Data Entry Work\Data Entry\20200827\CKA\4 D.JPG

**Answer: A**

## Question 6



The screenshot shows a web terminal interface with a dark blue header. On the left, there are two buttons: 'Readme' and 'Web Terminal'. On the right, the 'THE LINUX FOUNDATION' logo is displayed. The main area is a terminal window with a black background and white text. It displays a Kubernetes manifest for a Pod named 'nginx-kusc00101'. The manifest includes the following fields: 'apiVersion: v1', 'kind: Pod', 'metadata: name: nginx-kusc00101', 'spec: containers: - name: nginx, image: nginx, imagePullPolicy: IfNotPresent', and 'nodeSelector: disk: ssd'. At the bottom of the terminal, a status bar indicates '"disk.yaml" [New] 11L, 176C written'.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-kusc00101
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disk: ssd

"disk.yaml" [New] 11L, 176C written
```

**Options:**

A. solution

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

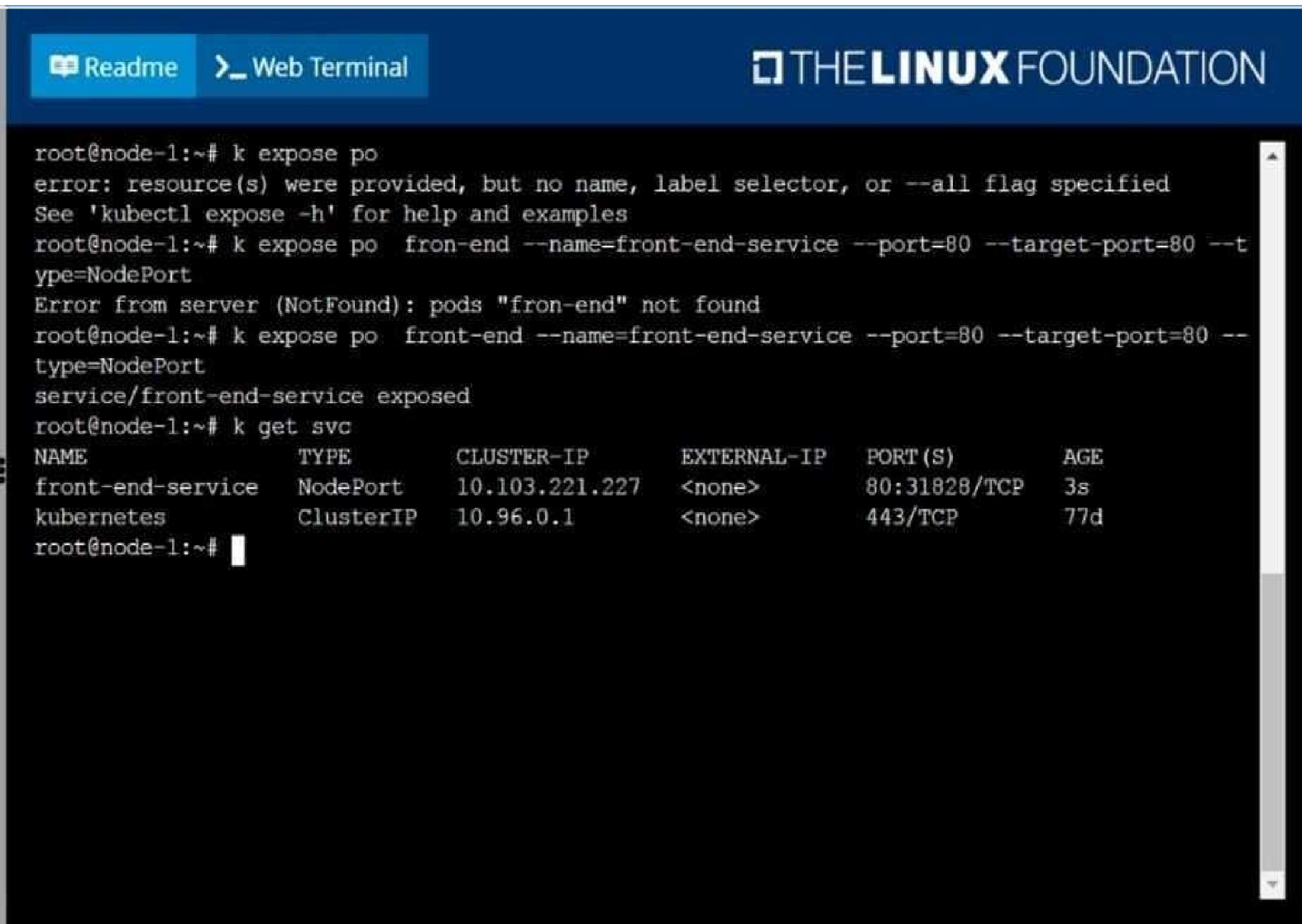
F:\Work\Data Entry Work\Data Entry\20200827\CKA\6 C.JPG

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

**Answer: A**

## Question 8

SIMULATION



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

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

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

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

**Options:**

A. solution

F:\Work\Data Entry Work\Data Entry\20200827\CKA\8 B.JPG

**Answer: A**

## Question 10

### SIMULATION

Create a pod as follows:

Name: mongo

Using Image: mongo

In a new Kubernetes namespace named: my-website

### Options:

[Readme](#)[Web Terminal](#)

F:\Work\Data Entry\Work\Data Entry\20200827\CKA\9. B. JPC

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

<https://www.certification-exam.com/en/pdf/linux-foundation-pdf/cka-pdf/>