Linux Foundation

CKA Exam

Certified Kubernetes Administrator (CKA) Program Exam

Product Questions: 122
Version: 4.0

Question: 1

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

Answer:

kubectl create namespace development
kubectl run nginx --image=nginx --restart=Never -n development

Question: 2

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: A

Question: 3

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

Answer: .

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

Question: 4

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

Answer:

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

Question: 5

CORRECT TEXT
Create a busybox pod that runs the command "env" and save the output to "envpod" file

Answer:

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

Question: 6

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

Answer:

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

Question: 7

CORRECT TEXT
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"

Question: 8

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

A. kubectl run nginx --image=nginx --restart=Never --env=var1=value1
then
kubectl exec -it nginx -- env
or
kubectl describe po nginx | grep value1
B. kubectl run nginx --image=nginx --restart=Never --env=var1=value1
then
kubectl exec -it nginx -- env
or
kubectl exec -it nginx -- sh -c 'echo $var1'
or
kubectl describe po nginx | grep value1

Answer: B

Question: 9

CORRECT TEXT

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

Answer:

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

Question: 10

CORRECT TEXT

Check the image version in pod without the describe command

Answer:

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

Question: 11

CORRECT TEXT

List the nginx pod with custom columns POD_NAME and POD_STATUS

Answer:

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

Question: 12

CORRECT TEXT
List all the pods sorted by name

Answer:

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

Question: 13

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
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

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
labels:
run: multi-container
name: multi-container
spec:
containers:
- image: nginx
name: nginx-container
- image: redis
name: consul-container
restartPolicy: Always

Answer: A

Question: 14

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

A. kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like "creationTimestamp: null" "dnsPolicy: ClusterFirst"
vim nginx-prod-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
env: prod
name: nginx-prod
spec:
containers:
- image: nginx
name: nginx-prod
restartPolicy: Always
kubectl create -f nginx-prod-pod.yaml
kubectl run --generator=run-pod/v1 --image=nginx --
labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
env: dev
name: nginx-dev
spec:
containers:
- image: nginx
name: nginx-dev
restartPolicy: Always
kubectl create -f nginx-prod-dev.yaml
Verify :
kubectl get po --show-labels
kubectl get po -l env=prod
kubectl get po -l env=dev
B. kubectl run --generator=run-pod/v1 --image=nginx -- labels=env=prod nginx-prod --dry-run -o yaml > nginx-prodpod.yaml Now, edit nginx-prod-pod.yaml file and remove entries like "creationTimestamp: null" "dnsPolicy: ClusterFirst"
vim nginx-prod-pod.yaml
apiVersion: v1
kind: Pod

metadata:
labels:
env: prod
name: nginx-prod
spec:
containers:
- image: nginx
name: nginx-prod
restartPolicy: Always
kubectl create -f nginx-prod-pod.yaml
kubectl run --generator=run-pod/v1 --image=nginx --
labels=env=dev nginx-dev --dry-run -o yaml > nginx-dev-pod.yaml
apiVersion: v1
kind: Pod
metadata:
- image: nginx
name: nginx-dev
restartPolicy: Always
kubectl create -f nginx-prod-dev.yaml
Verify :
kubectl get po --show-labels
kubectl get po -l env=dev

Answer: A

Question: 15

Get IP address of the pod – "nginx-dev"

A. Kubect1 get po -o wide
Using JsonPath
kubect1 get pods
.items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'
B. Kubect1 get po -o wide
Using JsonPath
kubect1 get pods -o=jsonpath='{range
.items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

Answer: B

Question: 16

CORRECT TEXT
Print pod name and start time to "/opt/pod-status" file

Answer:

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

Question: 17

CORRECT TEXT
Check the Image version of nginx-dev pod using jsonpath

Answer:

kubect1 get po nginx-dev -o
jsonpath='{.spec.containers[].image}{"\n"}'

Question: 18

CORRECT TEXT
Create a busybox pod and add “sleep 3600” command

Answer:

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

Question: 19

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

A. // 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
B. // 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=6
kubectl get po nginx --v=9

Answer: A

Question: 20

CORRECT TEXT
List the nginx pod with custom columns POD_NAME and POD_STATUS

Answer:

kubectl get po -o=custom-columns="POD_NAME:.metadata.name,

POD_STATUS:.status.containerStatuses[].state"

Question: 21

CORRECT TEXT
List all the pods sorted by name

Answer:

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

Question: 22

CORRECT TEXT
List all the pods sorted by created timestamp

Answer:

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

Question: 23

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

Answer:

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

Question: 24

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

A. kubect1 get pods -o wide
kubectl delete po
kubectl delete po “nginx-prod”

B. kubect1 get pods -o wide
kubectl delete po “nginx-dev”
kubectl delete po “nginx-prod”

Answer: B

Question: 25

CORRECT TEXT

Delete the pod without any delay (force delete)

Answer:

Kubect1 delete po “POD-NAME” --grace-period=0 –force

Question: 26

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:
- image: redis
name: redis
ports:
- containerPort: 6379
Rt restartPolicy: Always

B. 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

Answer: A

Question: 27

CORRECT TEXT
Create the nginx pod with version 1.17.4 and expose it on port 80

Answer:

kubectl run nginx --image=nginx:1.17.4 --restart=Never --
port=80

Question: 28

Change the Image version to 1.15-alpine for the pod you just created and verify the image version is updated.

A. Kubect1 set image pod/nginx nginx=nginx:1.15-alpine
kubect1 describe po nginx
// another way it will open vi editor and change the version
kubeclt edit po nginx
kubect1 describe po nginx
B. Kubect1 set image pod/nginx nginx=nginx:1.15-alpine
kubect1 describe po nginx
// another way it will open vi editor and change the version
kubect1 describe po nginx

Answer: A

Question: 29

CORRECT TEXT
Change the Image version back to 1.17.1 for the pod you just updated and observe the changes

Answer:

kubectl set image pod/nginx nginx=nginx:1.17.1
kubectl describe po nginx
kubectl get po nginx -w # watch it

Question: 30

Create a redis pod, and have it use a non-persistent storage
Note: In exam, you will have access to kubernetes.io site,
Refer : https://kubernetes.io/docs/tasks/configure-pod-container/configurevolume-storage/

A. apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
volumeMounts:
- name: redis-storage
mountPath: /data/redis
ports:
- containerPort: 6379

volumes:
- name: redis-storage
emptyDir: {}
B. apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
volumeMounts:
- containerPort: 6379
volumes:
- name: redis-storage
emptyDir: {}

Answer: A

Question: 31

Create a Pod with three busy box containers with commands “ls; sleep 3600;”, “echo Hello World; sleep 3600;” and “echo this is the third container; sleep 3600” respectively and check the status

A. // first create single container pod with dry run flag
kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- bin/sh -c "sleep 3600; ls" > multi-container.yaml
// edit the pod to following yaml and create it
apiVersion: v1
kind: Pod
metadata:
labels:
run: busybox
name: busybox
spec:
containers:
- args:
- bin/sh
- -c
- ls; sleep 3600
- echo Hello world; sleep 3600
image: busybox
name: busybox-container-2
- args:
- bin/sh
- -c
- echo this is third container; sleep 3600
image: busybox

name: busybox-container-3
restartPolicy: Always
// Verify
Kubectl get pods
B. // first create single container pod with dry run flag
kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- bin/sh -c "sleep 3600; ls" > multi-container.yaml
// edit the pod to following yaml and create it
apiVersion: v1
kind: Pod
metadata:
labels:
run: busybox
name: busybox
spec:
containers:
- args:
- bin/sh
- -c
- ls; sleep 3600
image: busybox
name: busybox-container-1
- args:
- bin/sh
- -c
- echo Hello world; sleep 3600
image: busybox
name: busybox-container-2
- args:
- bin/sh
- -c
- echo this is third container; sleep 3600
image: busybox
name: busybox-container-3
restartPolicy: Always
// Verify
Kubectl get pods

Answer: B

Question: 32

Check logs of each container that “busyboxpod-{1,2,3}”

A. kubectl logs busybox -c busybox-container-1
kubectl logs busybox -c busybox-container-2
kubectl logs busybox -c busybox-container-3
B. kubectl logs busybox -c busybox-container-1

kubectl logs busybox -c busybox-container-3
kubectl logs busybox -c busybox-container-3

Answer: A

Question: 33

Create a Pod with main container busybox and which executes this "while true; do echo 'Hi I am from Main container' >> /var/log/index.html; sleep 5; done" and with sidecar container with nginx image which exposes on port 80. Use emptyDir Volume and mount this volume on path /var/log for busybox and on path /usr/share/nginx/html for nginx container. Verify both containers are running.

A. // create an initial yaml file with this
kubectl run multi-cont-pod --image=busbox --restart=Never --dry-run -o yaml > multi-container.yaml
// edit the yml as below and create it
kubectl create -f multi-container.yaml
vim multi-container.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: multi-cont-pod
name: multi-cont-pod
spec:
volumes:
- name: var-logs
emptyDir: {}
containers:
- image: busybox
command: ["/bin/sh"]
args: ["-c", "while true; do echo 'Hi I am from Main container' >> /var/log/index.html; sleep 5;done"]
name: main-container
volumeMounts:
- name: var-logs
mountPath: /var/log
- image: nginx
name: sidecar-container
ports:
- containerPort: 80
volumeMounts:
- name: var-logs
mountPath: /usr/share/nginx/html
restartPolicy: Never

// Create Pod
kubectl apply -f multi-container.yaml
//Verify
kubectl get pods
B. // create an initial yaml file with this
kubectl run multi-cont-pod --image=busbox --restart=Never --
dry-run -o yaml > multi-container.yaml
// edit the yml as below and create it
kubectl create -f multi-container.yaml
vim multi-container.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: multi-cont-pod
name: multi-cont-pod
spec:
volumes:
- image: busybox
command: ["/bin/sh"]
args: ["-c", "while true; do echo 'Hi I am from Main
container' >> /var/log/index.html; sleep 5;done"]
name: main-container
volumeMounts:
- name: var-logs
mountPath: /var/log
- image: nginx
name: sidecar-container
ports:
mountPath: /usr/share/nginx/html
restartPolicy: Never
// Create Pod
kubectl apply -f multi-container.yaml
//Verify
kubectl get pods

Answer: A

Question: 34

CORRECT TEXT
Exec into both containers and verify that main.txt exist and

Answer:

NEED TO WRITE ANSWER FOR THIS

Question: 35

Create an nginx pod and set an env value as 'var1=val1'. Check the env value existence within the pod

A. kubectl run nginx --image=nginx --restart=Never --env=var1=val1
then
kubectl exec -it nginx -- env
or
kubectl exec -it nginx -- sh -c 'echo $var1'
or
kubectl describe po nginx | grep val1
or
kubectl run nginx --restart=Never --image=nginx --env=var1=val1
-it --rm – env
B. kubectl run nginx --image=nginx --restart=Never --env=var1=val1
then
kubectl exec -it nginx -- env
or
kubectl run nginx --restart=Never --image=nginx --env=var1=val1
-it --rm -- env

Answer: A

Question: 36

Create a pod with init container which create a file “test.txt”
in “workdir” directory. Main container should check a file
“test.txt” exists and execute sleep 9999 if the file exists.

A. // create an initial yaml file with this
kubectl run init-cont-pod --image=alpine --restart=Never --dry-run -o
yaml > init-cont-pod.yaml
// edit the yml as below and create it
vim init-cont-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: init-cont-pod
labels:
app: myapp
spec:
volumes:
- name: test-volume
emptyDir: {}
containers:

- name: main-container
image: busybox:1.28
command: ['sh', '-c', 'if [-f /workdir/test.txt]; then sleep
9999; fi']
volumeMounts:
- name: test-volume
mountPath: /workdir
initContainers:
- name: init-myservice
image: busybox:1.28
command: ['sh', '-c', "mkdir /workdir; echo >
/workdir/test.txt"]
volumeMounts:
- name: test-volume
mountPath: /workdir
// Create the pod
kubectl apply -f init-cont-pod.yaml
kubectl get pods
// Check Events by doing
kubectl describe po init-cont-pod
Init Containers:
init-myservice:
Container ID:
docker://ebdbf5fad1c95111d9b0e0e2e743c2e347c81b8d4eb5abcccdfe1dd74524
0d4f
Image: busybox:1.28
Image ID: dockerpullable://busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df
416dea4f41046e0f37d47
Port: <none>
Host Port: <none>
Command:
sh
-c
mkdir /workdir; echo > /workdir/test.txt
State: Terminated
Reason: Completed
B. // create an initial yaml file with this
kubectl run init-cont-pod --image=alpine --restart=Never --dry-run -o
yaml > init-cont-pod.yaml
// edit the yml as below and create it
vim init-cont-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: init-cont-pod
labels:
app: myapp
spec:

volumes:
- name: test-volume
emptyDir: {}
containers:
- name: main-container
image: busybox:1.28
command: ['sh', '-c', 'if [-f /workdir/test.txt]; then sleep
9999; fi']
volumeMounts:
image: busybox:1.28
command: ['sh', '-c', "mkdir /workdir; echo >
/workdir/test.txt"]
volumeMounts:
- name: test-volume
mountPath: /workdir
// Create the pod
kubectl apply -f init-cont-pod.yaml
kubectl get pods
// Check Events by doing
kubectl describe po init-cont-pod
Init Containers:
init-myservice:
Container ID:
docker://ebdbf5fad1c95111d9b0e0e2e743c2e347c81b8d4eb5abcccdfe1dd74524
0d4f
Image: busybox:1.28
Image ID: dockerpullable://busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df
416dea4f41046e0f37d47
Port: <none>
Host Port: <none>
Command:
sh
-c
mkdir /workdir; echo > /workdir/test.txt
State: Terminated
Reason: Completed

Answer: A

Question: 37

Create a pod with init container which waits for a service called “myservice” to be created. Once init container completes, the myapp-container should start and print a message “The app is running” and sleep for 3600 seconds.

A. vim multi-container-pod.yaml
apiVersion: v1
kind: Pod

metadata:
name: myapp-pod
labels:
app: myapp
spec:
containers:
- name: myapp-container
image: busybox:1.28
command: ['sh', '-c', 'echo The app is running! && sleep
3600']
initContainers:
- name: init-myservice
done"]
// Check whether service called “myservice” exists
kubectl get svc
Note: Pod will not start if service called “myservice” doesn’t
exist.
// Now, Create the pod
kubectl apply -f multi-container-pod.yaml
B. vim multi-container-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: myapp-pod
labels:
app: myapp
spec:
containers:
- name: myapp-container
image: busybox:1.28
command: ['sh', '-c', 'echo The app is running! && sleep
3600']
initContainers:
- name: init-myservice
image: busybox:1.28
command: ['sh', '-c', "until nslookup myservice.$(cat
/var/run/secrets/kubernetes.io/serviceaccount/namespace).s
vc.cluster.local; do echo waiting for myservice; sleep 2;
done"]
// Check whether service called “myservice” exists
kubectl get svc
Note: Pod will not start if service called “myservice” doesn’t
exist.
// Now, Create the pod
kubectl apply -f multi-container-pod.yaml

Answer: B

Question: 38

Create 5 nginx pods in which two of them is labeled env=prod and three of them is labeled env=dev

A. kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev3 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-prod1 --image=nginx --restart=Never --labels=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --labels=env=prod
B. kubectl run nginx-dev1 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --labels=env=dev
kubectl run nginx-prod1 --image=nginx --restart=Never --labels=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --labels=env=prod

Answer: A

Question: 39

CORRECT TEXT
Get the pods with label env=dev and output the labels

Answer:

kubectl get pods -l env=dev --show-labels

Question: 40

CORRECT TEXT
Get the pods with labels env=dev and env=prod and output the labels as well

Answer:

kubectl get pods -l 'env in (dev,prod)' --show-labels

Question: 41

CORRECT TEXT
Get all the pods with label “env”

Answer:

kubectl get pods -L env

Question: 42

CORRECT TEXT
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

Question: 43

CORRECT TEXT
Get list of all the nodes with labels

Answer:

kubectl get nodes --show-labels

Question: 44

Create a nginx pod that will be deployed to node with the label "gpu=true"

A. kubectl run nginx --image=nginx --restart=Always --dry-run -o yaml > nodeselector-pod.yaml
// add the nodeSelector like below and create the pod
kubectl apply -f nodeselector-pod.yaml
vim nodeselector-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
nodeSelector:
gpu: true
containers:
- image: nginx
name: nginx
restartPolicy: Always
kubectl apply -f nodeselector-pod.yaml
//Verify
kubectl get no –show-labels
kubectl get po
kubectl describe po nginx | grep Node-Selectors
B. kubectl run nginx --image=nginx --restart=Always --dry-run -o

yaml > nodeselector-pod.yaml
// add the nodeSelector like below and create the pod
kubectl apply -f nodeselector-pod.yaml
vim nodeselector-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
nodeSelector:
gpu: true
yaml
//Verify
kubectl get no –show-labels
kubectl get po
kubectl describe po nginx | grep Node-Selectors

Answer: A

Question: 45

Annotate the pod with name=webapp

A. kubectl annotate pod nginx-dev-pod name=webapp
kubectl annotate pod nginx-prod-pod name=webapp
// Verify
kubectl describe po nginx-dev-pod | grep -i annotations
B. kubectl annotate pod nginx-dev-pod name=webapp
kubectl annotate pod nginx-prod-pod name=webapp
// Verify
kubectl describe po nginx-dev-pod | grep -i annotations
kubectl describe po nginx-prod-pod | grep -i annotations

Answer: B

Question: 46

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

Question: 47

Get the list of pods of webapp deployment

A. // Get the label of the deployment
kubectl get deploy --show-labels
kubectl get pods -l app=webapp
B. // Get the label of the deployment
kubectl get deploy --show-labels
// Get the pods with that label
kubectl get pods -l app=webapp

Answer: B

Question: 48

CORRECT TEXT
Scale the deployment from 5 replicas to 20 replicas and verify

Answer:

kubectl scale deploy webapp --replicas=20
kubectl get deploy webapp
kubectl get po -l app=webapp

Question: 49

CORRECT TEXT
Get the deployment rollout status

Answer:

kubectl rollout status deploy webapp

Question: 50

Deployment

a. Create a deployment of webapp with image nginx:1.17.1 with
container port 80 and verify the image version
A. // Create initial YAML file with –dry-run option
kubectl create deploy webapp --image=nginx:1.17.1 --dryrun=client -o yaml > webapp.yaml
vim webapp.yaml
apiVersion: apps/v1
kind: Deployment

metadata:
labels:
app: webapp
name: webapp
spec:
replicas: 1
selector:
matchLabels:
app: webapp
template:
metadata:
labels:
app: webapp
spec:
containers:
- image: nginx:1.17.1
name: nginx
kubectl create -f webapp.yaml –record=true
//Verify Image Version
kubectl describe deploy webapp | grep -i "Image"
Using JsonPath
kubectl get deploy -o=jsonpath='{range.items [*]}{.[*]}
{.metadata.name}{"\t"}{.spec.template.spec.containers[*].i
mage}{"\n"}'
B. // Create initial YAML file with –dry-run option
kubectl create deploy webapp --image=nginx:1.17.1 --dryrun=client -o yaml > webapp.yaml
vim webapp.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: webapp
name: webapp
spec:
replicas: 1
containers:
- image: nginx:1.17.1
name: nginx
kubectl create -f webapp.yaml –record=true
//Verify Image Version
kubectl describe deploy webapp | grep -i "Image"
Using JsonPath
kubectl get deploy -o=jsonpath='{range.items [*]}{.[*]}
{.metadata.name}{"\t"}{.spec.template.spec.containers[*].i
mage}{"\n"}'

Answer: A

Question: 51

Update the deployment with the image version 1.17.4 and verify

A. kubectl set image deploy/webapp nginx=nginx:1.17.4
//Verify
kubectl describe deploy webapp | grep Image
kubectl get deploy -o=jsonpath='{range.items [*]}{.[*]}
{.metadata.name}{"\t"}{.spec.template.spec.containers[*].i
mage}{"\n"}'
B. kubectl set image deploy/webapp nginx=nginx:1.17.4
//Verify
kubectl describe deploy webapp | grep Image
kubectl get deploy -
{.metadata.name}{"\t"}{.spec.template.spec.containers[*].i
mage}{"\n"}'

Answer: A

Question: 52

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 -l app=webapp
kubectl get po -l app=webapp
B. kubectl rollout history deploy webapp
kubectl get deploy webapp --show-labels
kubectl get rs -
kubectl get po -l app=webapp

Answer: A

Question: 53

CORRECT TEXT
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

Question: 54

CORRECT TEXT
Update the deployment with the image version 1.16.1 and verify the image and check the rollout history

Answer:

kubectl set image deploy/webapp nginx=nginx:1.16.1
kubectl describe deploy webapp | grep Image
kubectl rollout history deploy webapp

Question: 55

CORRECT TEXT
Check the history of deployment

Answer:

kubectl rollout history deployment webapp

Question: 56

CORRECT TEXT
Undo the deployment with the previous version and verify everything is Ok

Answer:

kubectl rollout undo deploy webapp
kubectl rollout status deploy webapp
kubectl get pods

Question: 57

Undo/Rollback deployment to specific revision “1”

A. // Check Deployment History
kubectl rollout history deployment webapp
//Rollback to particular revision
kubectl rollout undo deploymet webapp --to-revision=1

B. // Check Deployment History
kubectl rollout history deployment webapp
kubectl rollout undo deploymet webapp --to-revision=1

Answer: A

Question: 58

CORRECT TEXT
Check the history of the specific revision of that deployment

Answer:

kubectl rollout history deploy webapp --revision=3

Question: 59

CORRECT TEXT
Pause the rollout of the deployment

Answer:

kubectl rollout pause deploy webapp

Question: 60

CORRECT TEXT
Resume the rollout of the deployment

Answer:

kubectl rollout resume deploy webapp

Question: 61

CORRECT TEXT
Scale the deployment to 5 replicas

Answer:

kubectl scale deployment webapp –replicas=5
//Verify
kubectl get deploy
kubectl get po,rs

Question: 62

CORRECT TEXT
Scale down the deployment to 1 replica

Answer:

kubectl scale deployment webapp –replicas=1
//Verify
kubectl get deploy
kubectl get po,rs

Question: 63

CORRECT TEXT
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

Question: 64

CORRECT TEXT
Clean the cluster by deleting deployment and hpa you just created

Answer:

kubectl delete deploy webapp
kubectl delete hpa webapp

Question: 65

Create a deployment named “myapp” that having 2 replicas with nginx image and expose deployment as service named “myservice”

A. // Create a YAML Template
kubectl create deploy myapp --image=nginx --dry-run -o yaml > myapp.yaml
//Update replicas=2 in myapp.yaml file
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: myapp
name: myapp
spec:
replicas: 2
selector:
matchLabels:
app: myapp
template:
metadata:
labels:
app: myapp
spec:
containers:
- image: nginx
name: nginx
// Create deployment
kubectl create -f myapp.yaml
// Creating YAML template for service

kubectl expose deployment myapp --type=ClusterIP --port=80 --
target-port=80 --name=myservice --dry-run -o yaml >
myservice.yaml
YAML File:
apiVersion: v1
kind: Service
metadata:
labels:
app: myapp
name: myservice
spec:
ports:
- port: 80
protocol: TCP
targetPort: 80
selector:
app: myapp
type: ClusterIP
kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S)
AGE
kubernetes ClusterIP 10.2.0.1 <none> 443/TCP
158d
myservice ClusterIP 10.2.96.175 <none> 80/TCP
40s
B. // Create a YAML Template
kubectl create deploy myapp --image=nginx --dry-run -o yaml >
myapp.yaml
//Update replicas=2 in myapp.yaml file
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: myapp
name: myapp
spec:
replicas: 2
selector:
matchLabels:
app: myapp
template:
metadata:
labels:
app: myapp
spec:
containers:
- image: nginx
name: nginx

// Create deployment
kubectl create -f myapp.yaml
// Creating YAML template for service
kubectl expose deployment myapp --type=ClusterIP --port=60 --
target-port=60 --name=myservice --dry-run -o yaml >
myservice.yaml
YAML File:
apiVersion: v1
kind: Service
metadata:
labels:
app: myapp
name: myservice
spec:
ports:
- port: 60
protocol: TCP
targetPort: 80
selector:
app: myapp
type: ClusterIP
kubectl get svc
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S)
AGE
kubernetes ClusterIP 10.2.0.1 <none> 443/TCP
158d
myservice ClusterIP 10.2.96.175 <none> 80/TCP
40s

Answer: A

Question: 66

Create the deployment redis with image=redis and expose it with “NodePort” service redis-service

A. kubectl create deploy redis --image=redis --dry-run -o yaml >
redis-deploy.yaml
Edit redis-deploy.yaml file
vim redis-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: redis
name: redis
spec:
replicas: 1
selector:

matchLabels:
app: redis
template:
metadata:
labels:
app: redis
spec:
containers:
- image: redis
name: redis
//Creating Service
kubectl expose deploy redis --type=NodePort --port=6379 --
target-port=6379 --name redis-service
// Verify
kubectl get deploy,svc
B. kubectl create deploy redis --image=redis --dry-run -o yaml >
redis-deploy.yaml
Edit redis-deploy.yaml file
name: redis
spec:
replicas: 1
selector:
matchLabels:
app: redis
template:
metadata:
labels:
app: redis
spec:
containers:
- image: redis
name: redis
//Creating Service
kubectl expose deploy redis --type=NodePort --port=6379 --
target-port=6379 --name redis-service
// Verify
kubectl get deploy,svc

Answer: A

Question: 67

Get the DNS records for the service and pods for the deployment redis and the put the value in /tmp/dnsrecordpod and /tmp/dnsrecord-service

A. // Get Pod Ip
kubectl get po -o wide
// Get Service Name

kubectl get svc
// Create a temporary pod and execute nslookup command
Note: POD IP format should be a-b-c-d and not a.b.c.d
kubectl run busybox --image=busybox:1.28 --restart=Never --rm -it -- nslookup 192-168-0-69.default.pod > /tmp/dnsrecord-pod
kubectl run busybox1 --image=busybox:1.26 --restart=Never --rm -it -- nslookup redis-service > /tmp/dnsrecordservice
//Verify
cat /tmp/dnsrecord-pod
Server: 10.2.8.10
Address 1: 10.2.0.10 kube-dns.kube system.svc.cluster.local
Name: 192-168-0-69.default.pod
Address 1: 192.168.0.69 192-166-0-69.redis service.default.svc.cluster.local
cat /tmp/dnsrecord-pod
Server: 10.2.0.10
Address 1: 10.2.0.10 kube-dns.kube system.svc.cluster.local
Name: 192-168-0-69.default.pod
Address 1: 192.168.0.69 192-168-0-69.redis service.default.svc.cluster.local
B. // Get Pod Ip
kubectl get po -o wide
// Get Service Name
kubectl get svc
// Create a temporary pod and execute nslookup command
Note: POD IP format should be a-b-c-d and not a.b.c.d
kubectl run busybox --image=busybox:1.28 --restart=Never --rm -it -- nslookup 192-168-0-69.default.pod > /tmp/dnsrecord-pod
kubectl run busybox1 --image=busybox:1.28 --restart=Never --rm -it -- nslookup redis-service > /tmp/dnsrecordservice
//Verify
cat /tmp/dnsrecord-pod
Server: 10.2.0.10
Address 1: 10.2.0.10 kube-dns.kube system.svc.cluster.local
Name: 192-168-0-69.default.pod
Address 1: 192.168.0.69 192-168-0-69.redis service.default.svc.cluster.local
cat /tmp/dnsrecord-pod
Server: 10.2.0.10
Address 1: 10.2.0.10 kube-dns.kube system.svc.cluster.local
Name: 192-168-0-69.default.pod
Address 1: 192.168.0.69 192-168-0-69.redis

service.default.svc.cluster.local

Answer: B

Question: 68

CORRECT TEXT
List all the pods that are serviced by the service "webservice" and copy the output in /opt/$USER/webservice.targets
Note: You need to list the endpoints

Answer:

kubectl descrive svc webservice | grep -i "Endpoints" >
/opt/$USER/webservice.targets
kubectl get endpoints webservice >
/opt/$USER/webservice.targets

Question: 69

Create a Job with an image node which prints node version and
verifies there is a pod created for this job

A. kubectl create job nodeversion --image=node -- node -v
kubectl get job -w
kubectl get pod
YAML File:
apiVersion: batch/v1
kind: Job
metadata:
labels:
job-name: nodeversion
name: nodeversion
spec:
completions: 1
parallelism: 1
selector:
matchLabels:
job-name: nodeversion
template:
metadata:
labels:
job-name: nodeversion
spec:
containers:
- command:
- node
- -v
image: node

imagePullPolicy: Always
name: nodeversion
restartPolicy: Never
B. kubectl create job nodeversion --image=node -- node -v
kubectl get job -w
kubectl get pod
YAML File:
apiVersion: batch/v1
kind: Job
metadata:
labels:
job-name: nodeversion
name: nodeversion
spec:
completions: 1
parallelism: 1
labels:
job-name: nodeversion
spec:
containers:
- command:
- node
- -v
image: node
imagePullPolicy: Always
name: nodeversion
restartPolicy: Never

Answer: A

Question: 70

Create a job named “hello-job” with the image busybox which echos “Hello I’m running job”

A. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I’m running job" > hello-job.yaml
kubectl create -f hello-job.yaml
//Verify Job
kubectl get job
kubectl get po
kubectl logs hello-job-*
B. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I’m running job" > hello-job.yaml
kubectl create -f hello-job.yaml
//Verify Job
kubectl get po
kubectl logs hello-job-*

Answer: A

Question: 71

Modify “hello-job” and make it run 10 times one after one and 5 times parallelism: 5

A. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I am from job" > hello-job.yaml
// edit the yaml file to add completions: 16 and
kubectl create -f hello-job.yaml
YAML File:
apiVersion: batch/v1
kind: Job
metadata:
name: hello-job
spec:
completions: 16
parallelism: 5
template:
metadata:
spec:
containers:
- command:
- echo
- Hello I am from job
image: busybox
name: hello-job
restartPolicy: Never
B. kubectl create job hello-job --image=busybox --dry-run -o yaml
-- echo "Hello I am from job" > hello-job.yaml
// edit the yaml file to add completions: 10 and
kubectl create -f hello-job.yaml
YAML File:
apiVersion: batch/v1
kind: Job
metadata:
name: hello-job
spec:
completions: 10
parallelism: 5
template:
metadata:
spec:
containers:
- command:
- echo
- Hello I am from job
image: busybox

name: hello-job
restartPolicy: Never

Answer: B

Question: 72

CORRECT TEXT
Watch the job that runs 10 times one by one and verify 10 pods are created and delete those after it's completed

Answer:

kubectl get job -w
kubectl get po
kubectl delete job hello-job

Question: 73

Create a Cronjob with busybox image that prints date and hello
from kubernetes cluster message for every minute

A. CronJob Syntax:
* --> Minute
* --> Hours
* --> Day of The Month
* --> Month
* --> Day of the Week
*/1 * * * * --> Execute a command every one minutes.
vim date-job.yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
name: date-job
spec:
schedule: "*/1 * * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: hello
image: busybox
args:
- /bin/sh
- -c
- date; echo Hello from the Kubernetes cluster
restartPolicy: OnFailure
kubectl apply -f date-job.yaml
//Verify

kubectl get cj date-job -o yaml
B. CronJob Syntax:
* --> Minute
* --> Hours
* --> Day of The Month
* --> Month
* --> Day of the Week
*/1 * * * * --> Execute a command every one minutes.
vim date-job.yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
name: date-job
spec:
schedule: "*/1 * * * *"
jobTemplate:
spec:
template:
- /bin/sh
- -c
- date; echo Hello from the Kubernetes cluster
restartPolicy: OnFailure
kubectl apply -f date-job.yaml
//Verify
kubectl get cj date-job -o yaml

Answer: A

Question: 74

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

Question: 75

CORRECT TEXT
Get list of persistent volumes and persistent volume claim in the cluster

Answer:

kubectl get pv
kubectl get pvc

Question: 76

Create a redis pod, and have it use a non-persistent storage (volume that lasts for the lifetime of the Pod)

A. vim redis-pod-vol.yaml
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:

containers:
- name: redis
image: redis
ports:
- containerPort: 6379
volumeMounts:
- mountPath: /data
name: redis-storage
volumes:
- name: redis-storage
emptyDir: {}
kubectl apply -f redis-pod-vol.yaml

B. vim redis-pod-vol.yaml
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
ports:
- containerPort: 6679
volumeMounts:
- mountPath: /data
name: redis-storage
volumes:
- name: redis-storage
emptyDir: {}
kubectl apply -f redis-pod-vol.yaml

Answer: A

Question: 77

Create a hostPath PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteOnce, storageClassName manual, and volume at /mnt/data and verify

A. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: task-pv-volume
labels:
type: local
spec:
storageClassName: ""

capacity:
storage: 5Gi
accessModes:
- ReadWriteOnce
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
NAME CAPACITY ACCESS
MODES RECLAIM POLICY STATUS CLAIM
STORAGECLASS REASON AGE
task-pv-volume 5Gi RWO
Retain Available
3s
B. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: task-pv-volume
labels:
type: local
spec:
storageClassName: ""
capacity:
storage: 5Gi
accessModes:
- ReadWriteOnce
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
NAME CAPACITY ACCESS
MODES RECLAIM POLICY STATUS CLAIM
STORAGECLASS REASON AGE
task-pv-volume 4Gi RWO
Retain Available
8s

Answer: A

Question: 78

Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify status is Bound

A. vim task-pv-claim.yaml

apiVersion: v2
kind: PersistentVolumeClaim
metadata:
name: task-pv-claim
spec:
storageClassName: “”
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 4Gi
kubectl apply -f task-pv-claim.yaml
//Verify
kubectl get pv
NAME CAPACITY ACCESS
MODES RECLAIM POLICY STATUS CLAIM
STORAGECLASS REASON AGE
task-pv-volume 4Gi RWO
Retain Bound default/task-pv-claim
6m16s
kubectl get pvc
NAME STATUS VOLUME
CAPACITY ACCESS MODES STORAGECLASS AGE
task-pv-claim Bound task-pv-volume
5Gi RWO 6s
B. vim task-pv-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: task-pv-claim
spec:
storageClassName: “”
accessModes:
- ReadWriteOnce
resources:
requests:
storage: 3Gi
kubectl apply -f task-pv-claim.yaml
//Verify
kubectl get pv
NAME CAPACITY ACCESS
MODES RECLAIM POLICY STATUS CLAIM
STORAGECLASS REASON AGE
task-pv-volume 5Gi RWO
Retain Bound default/task-pv-claim
6m16s
kubectl get pvc
NAME STATUS VOLUME

CAPACITY ACCESS MODES STORAGECLASS AGE
task-pv-claim Bound task-pv-volume
5Gi RWO 6s

Answer: B

Question: 79

CORRECT TEXT
Delete persistent volume and persistent volume claim

Answer:

kubectl delete pvc task-pv-claim
kubectl delete pv task-pv-volume
// Verify
Kubectl get pv,pvc

Question: 80

Create a redis pod named “test-redis” and exec into that pod and create a file named "test-file.txt" with the text ‘This is called the test file’ in the path /data/redis and open another tab and exec again with the same pod and verifies file exist in the same path.

A. vim test-redis.yaml
apiVersion: v1
kind: Pod
metadata:
name: test-redis
spec:
containers:
- name: redis
image: redis
ports:
- containerPort: 6379
volumeMounts:
- mountPath: /data/redis
name: redis-storage
volumes:
- name: redis-storage
emptyDir: {}
kubectl apply -f redis-pod-vol.yaml
// first terminal
kubectl exec -it test-redis /bin/sh
cd /data/redis
echo 'This is called the test file' > file.txt
//open another tab
kubectl exec -it test-redis /bin/sh

cat /data/redis/file.txt
B. vim test-redis.yaml
apiVersion: v1
kind: Pod
metadata:
name: test-redis
spec:
containers:
- name: redis
image: redis
ports:
- containerPort: 6379
volumeMounts:
- mountPath: /data/redis
name: redis-storage
volumes:
kubectl exec -it test-redis /bin/sh
cd /data/redis
echo 'This is called the test file' > file.txt
//open another tab
kubectl exec -it test-redis /bin/sh
cat /data/redis/file.txt

Answer: A

Question: 81

CORRECT TEXT
Delete the above pod and create again from the same yaml file and verifies there is no "test-file.txt" in the path /data/redis (Since non-persistent storage "emptyDir" is used).

Answer:

kubectl delete pod test-redis
kubectl create -f test-redis.yaml
kubectl exec -it test-redis /bin/sh
cat /data/redis/file.txt // file doesn't exist

Question: 82

Create PersistentVolume named task-pv-volume with storage 10Gi, access modes ReadWriteMany, storageClassName manual, and volume at /mnt/data and Create a PersistentVolumeClaim of at least 3Gi storage and access mode ReadWriteOnce and verify

A. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:

name: task-pv-volume
labels:
type: local
spec:
storageClassName: manual
capacity:
storage: 10Gi
accessModes:
- ReadWriteMany
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
vim task-pvc-volume.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: task-pv-claim
spec:
storageClassName: manual
accessModes:
- ReadWriteMany
resources:
requests:
storage: 3Gi
kubectl apply -f task-pvc-volume.yaml
//Verify
Kuk kubectl get pvc
B. vim task-pv-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
name: task-pv-volume
labels:
type: local
spec:
storageClassName: manual
capacity:
storage: 10Gi
accessModes:
- ReadWriteMany
hostPath:
path: "/mnt/data"
kubectl apply -f task-pv-volume.yaml
//Verify
kubectl get pv
vim task-pvc-volume.yaml

apiVersion: v1
- ReadWriteMany
resources:
requests:
storage: 3Gi
kubectl apply -f task-pvc-volume.yaml
//Verify
Kuk kubectl get pvc

Answer: A

Question: 83

Create an nginx pod with containerPort 80 and with a PersistentVolumeClaim "task-pv-claim" and has a mouth path "/usr/share/nginx/html"

A. vim nginx-pvc-pod.yaml
apiVersion: v1
kind: Pod
metadata:
name: task-pv-pod
spec:
volumes:
- name: task-pv-storage
persistentVolumeClaim:
claimName: task-pv-claim
containers:
- name: task-pv-container
image: nginx
ports:
- containerPort: 60
name: "http"
volumeMounts:
- mountPath: "/usr/share/nginx/html"
name: task-pv-storage
kubectl apply -f nginx-pvc-pod.yaml
// Verify
kubectl describe po “POD-Name” | grep -i volumes –A4
Volumes:
task-pv-storage:
Type: PersistentVolumeClaim (a reference to a
PersistentVolumeClaim in the same namespace)
ClaimName: task-pv-claim
ReadOnly: false
B. vim nginx-pvc-pod.yaml
apiVersion: v1
kind: Pod
metadata:

```
name: task-pv-pod
spec:
volumes:
- name: task-pv-storage
persistentVolumeClaim:
claimName: task-pv-claim
containers:
- name: task-pv-container
image: nginx
ports:
- containerPort: 80
name: "http"
volumeMounts:
- mountPath: "/usr/share/nginx/html"
name: task-pv-storage
kubectl apply -f nginx-pvc-pod.yaml
// Verify
kubectl describe po “POD-Name” | grep -i volumes -A5
Volumes:
task-pv-storage:
Type: PersistentVolumeClaim (a reference to a
PersistentVolumeClaim in the same namespace)
ClaimName: task-pv-claim
ReadOnly: false
```

Answer: B

Question: 84

CORRECT TEXT
Get list of PVs and order by size and write to file - /opt/pvlist.txt

Answer:

```
kubectl get pv --sort-by=.spec.capacity.storage > /opt/pvlist.txt
```

Question: 85

CORRECT TEXT
List all configmap and secrets in the cluster in all namespace and write it to a file /opt/configmap-secret

Answer:

```
kubectl get configmap,secrets --all-namespaces >
/opt/configmap-secret
// Verify
Cat /opt/configmap-secret
```

Question: 86

Create a configmap called myconfigmap with literal value
appname=myapp

A. kubectl create cm myconfigmap --from-literal=appname=myapp
// Verify
kubectl get cm -o yaml
(or)
kubectl describe cm

B. kubectl create cm myconfigmap --from-literal=appname=myapp
// Verify
(or)
kubectl describe cm

Answer: A

Question: 87

Create a file called “config.txt” with two values key1=value1
and key2=value2. Then create a configmap named “keyvalcfgmap” andread data from the file “config.txt” and verify that configmap is
created correctly

A. cat >> config.txt << EOF
key1=value1
key2=value2
EOF
kubectl create cm keyvalcfgmap --from-file=config.txt
//Verify
kubectl get cm keyvalcfgmap -o yaml

B. cat >> config.txt << EOF
key1=value1
key2=value2
EOF
cat config.txt
// Create configmap from "config.txt" file
kubectl create cm keyvalcfgmap --from-file=config.txt
//Verify
kubectl get cm keyvalcfgmap -o yaml

Answer: B

Question: 88

Create an nginx pod and load environment values from the above configmap "keyvalcfgmap" and exec into the pod and verify the environment variables and delete the pod

A. // first run this command to save the pod yaml
kubectl run nginx --image=nginx --restart=Always --dry-run -o
yaml > nginx-pod.yml
// edit the yml to below file and create
vim nginx-pod.yml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
envFrom:
- configMapRef:
name: keyvalcfgmap
restartPolicy: Always
kubectl apply -f nginx-pod.yml
// verify
kubectl exec -it nginx -- env
kubectl delete po nginx
B. // first run this command to save the pod yaml
kubectl run nginx --image=nginx --restart=Always --dry-run -o
yaml > nginx-pod.yml
// edit the yml to below file and create
vim nginx-pod.yml
apiVersion: v1
name: nginx
envFrom:
- configMapRef:
name: keyvalcfgmap
restartPolicy: Always
kubectl apply -f nginx-pod.yml
// verify
kubectl exec -it nginx -- env
kubectl delete po nginx

Answer: A

Question: 89

Create a redis pod and mount “redis-config” as “redis.conf”
inside redis container, name the config volume as “redis-volume”
redis-config path - /opt/redis-config

A. 1
B. 2
C. Pending

Answer: C

Question: 90

// Create a configmap
kubectl create configmap redis-config --from-file=/opt/redisconfig
// Verify
kubectl get configmap redis-config -o yaml
// first run this command to save the pod yml
kubectl run redis-pod --image=redis --restart=Always --dry-run
-o yaml > redis-pod.yml
// edit the yml to below file and create
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
containers:
- name: redis
image: redis
env:
- name: MASTER
value: "true"
ports:
- containerPort: 6379
volumeMounts:
- mountPath: /redis-master-data
name: data
- mountPath: /redis-master
name: config
volumes:
- name: data
emptyDir: {}
- name: config
configMap:
name: example-redis-config

A. items:
- key: redis-config
path: redis.conf
cf
kk kubectl apply -f redis-pod.yml
// // Verify
K kubectl exec -it redis – cat /redis-master-data/redis.conf
B. items:
- key: redis-config
path: redis.conf
cf
// // Verify
K kubectl exec -it redis – cat /redis-master-data/redis.conf

Answer: A

Question: 91

Create a configmap called cfgvolume with values var1=val1,
var2=val2 and create an nginx pod with volume nginx-volume which
reads data from this configmap cfgvolume and put it on the path
/etc/cfg

A. // first create a configmap cfgvolume
kubectl create cm cfgvolume --from-literal=var1=val1 --fromliteral=var2=val2
// verify the configmap
kubectl describe cm cfgvolume
// create the config map
kubectl create -f nginx-volume.yml
vim nginx-configmap-pod.yaml
apiVersion: v1
kind: Pod
- name: nginx-volume
configMap:
name: cfgvolume
containers:
- image: nginx
name: nginx
volumeMounts:
- name: nginx-volume
mountPath: /etc/cfg
restartPolicy: Always
k kubectl apply -f nginx-configmap-pod.yaml
/ // Verify
// exec into the pod
kubectl exec -it nginx -- /bin/sh
// check the path

cd /etc/cfg
B. // first create a configmap cfgvolume
kubectl create cm cfgvolume --from-literal=var1=val1 --fromliteral=var2=val2
// verify the configmap
kubectl describe cm cfgvolume
// create the config map
kubectl create -f nginx-volume.yml
vim nginx-configmap-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
volumes:
- name: nginx-volume
configMap:
name: cfgvolume
containers:
- image: nginx
name: nginx
volumeMounts:
- name: nginx-volume
mountPath: /etc/cfg
restartPolicy: Always
k kubectl apply -f nginx-configmap-pod.yaml
/ // Verify
// exec into the pod
kubectl exec -it nginx -- /bin/sh
// check the path
cd /etc/cfg

Answer: B

Question: 92

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 --all-namespaces
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 generic my-secret -o yaml

Answer: A

Question: 93

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
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
valueFrom:
secretKeyRef:
name: my-secret
key: username
restartPolicy: Never

kubectl create -f nginx.yml
//Verify
kubectl exec -it nginx – env

Answer: B

Question: 94

Create an nginx pod which loads the secret as environment variables

A. // create a yml file
kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx.yml
// add env section below and create
vim nginx.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
envFrom:
- secretRef:
name: my-secret
restartPolicy: Never
kubectl apply -f nginx.yaml
//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
vim nginx.yaml
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
envFrom:
- secretRef:
name: my-secret
restartPolicy: Never
kubectl apply -f nginx.yaml

//verify
kubectl exec -it nginx – env

Answer: A

Question: 95

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

Question: 96

Create a busybox pod which executes this command sleep 3600 with the service account admin and verify

A. kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- /bin/sh -c "sleep 3600" > busybox.yml
// Edit busybox.yaml file
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
run: busybox
name: busybox
spec:
serviceAccountName: admin
containers:
- args:
- /bin/sh
- -c
- sleep 3600
image: busybox
name: busybox
restartPolicy: Always
// verify

K kubectl describe po busybox
B. kubectl run busybox --image=busybox --restart=Always --dry-run
-o yaml -- /bin/sh -c "sleep 3600" > busybox.yml
// Edit busybox.yaml file
apiVersion: v1
kind: Pod
metadata:
creationTimestamp: null
labels:
run: busybox
name: busybox
spec:
serviceAccountName: admin
containers:
- args:
- /bin/sh
- -c
- sleep 3800
image: busybox
name: busybox
restartPolicy: Always
// verify
K kubectl describe po busybox

Answer: A

Question: 97

Create an nginx pod with container Port 80 and it should only receive traffic only it checks the endpoint / on port 80 and verify and delete the pod.

A. kubectl run nginx --image=nginx --restart=Never --port=80 --
dry-run -o yaml > nginx-pod.yaml
// add the readinessProbe section and create
vim nginx-pod.yaml
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
ports:
- containerPort: 60
readinessProbe:
httpGet:
path: /
port: 60
restartPolicy: Never

kubectl apply -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx
B. kubectl run nginx --image=nginx --restart=Never --port=80 --
dry-run -o yaml > nginx-pod.yaml
// add the readinessProbe section and create
vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
ports:
- containerPort: 80
readinessProbe:
httpGet:
path: /
port: 80
restartPolicy: Never
kubectl apply -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx

Answer: B

Question: 98

List all the events sorted by timestamp and put them into file.log and verify

A. kubectl get events --sort-by=.metadata.creationTimestamp
kubectl get events --sort-by=.metadata.creationTimestamp >
test-file.log
cat test-file.log
B. kubectl get events --sort-by=.metadata.creationTimestamp
// putting them into file.log
kubectl get events --sort-by=.metadata.creationTimestamp >
cat test-file.log
C. kubectl get events --sort-by=.metadata.creationTimestamp
// putting them into file.log
kubectl get events --sort-by=.metadata.creationTimestamp >
test-file.log

cat test-file.log

Answer: C

Question: 99

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 -3
// putting into file
kubectl top pod --all-namespaces | sort --reverse --key 3 --
numeric | head -3 > 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 -8
// putting into file
kubectl top pod --all-namespaces | sort --reverse --key 6 --
numeric | head -6 > cpu-usage.txt
// verify
cat cpu-usage.txt

Answer: A

Question: 100

CORRECT TEXT
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

Question: 101

Create a Pod nginx and specify a CPU request and a CPU limit of 0.5 and 1 respectively.

A. // create a yml file
kubectl run nginx-pod --image=nginx --restart=Never --dry-run -
o yaml > nginx-pod.yml
// add the resources section and create
vim nginx-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
resources:
requests:
cpu: "0.5"
limits:
cpu: "1"
restartPolicy: Always
kubectl apply -f nginx-pod.yaml
// verify
kubectl top pod
```

B.

```
// create a yml file
kubectl run nginx-pod --image=nginx --restart=Never --dry-run -
o yaml > nginx-pod.yml
// add the resources section and create
vim nginx-pod.yaml
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
resources:
requests:
cpu: "0.4"
limits:
cpu: "1"
restartPolicy: Always
kubectl apply -f nginx-pod.yaml
// verify
kubectl top pod
```

Answer: A

Question: 102

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
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
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-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

Answer: B

Question: 103

Set CPU and memory requests and limits for existing pod name "nginx-prod".
Set requests for CPU and Memory as 100m and 256Mi respectively
Set limits for CPU and Memory as 200m and 512Mi respectively

A. kubectl get po
kubectl set resources po nginx-prod --limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi
//Verify
kubectl top po
kubectl describe po nginx-prod
B. kubectl get po
kubectl set resources po nginx-prod --limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi
//Verify
kubectl describe po nginx-prod

Answer: A

Question: 104

Create an nginx pod with containerPort 80 and it should check the pod running at endpoint / healthz on port 80 and verify and delete the pod.

A. kubectl run nginx --image=nginx --restart=Always --port=80 --dry-run -o yaml > nginx-pod.yaml
// add the livenessProbe section and create
apiVersion: v1
kind: Pod
metadata:
labels:
run: nginx
name: nginx
spec:
containers:
- image: nginx
name: nginx
ports:
- containerPort: 80
livenessProbe:
httpGet:
path: /healthz
port: 80
restartPolicy: Always
kubectl create -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness

kubectl delete po nginx
B. kubectl run nginx --image=nginx --restart=Always --port=80 --
dry-run -o yaml > nginx-pod.yaml
// add the livenessProbe section and create
apiVersion: v1
kind: Pod
metadata:
labels:
containers:
- image: nginx
name: nginx
ports:
- containerPort: 60
livenessProbe:
httpGet:
path: /healthz
port: 60
restartPolicy: Always
kubectl create -f nginx-pod.yaml
// verify
kubectl describe pod nginx | grep -i readiness
kubectl delete po nginx

Answer: A

Question: 105

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: A

Question: 106

Allow traffic from all the pods in "web" namespace and from pods with label "type=monitoring" to the pods matching label "app: db"

A. kubectl create namespace web
kubectl label namespace/web app=web
vim web-allow-all-ns-monitoring.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: web-allow-all-ns-monitoring
namespace: default
spec:
podSelector:
matchLabels:
app: db
ingress:
- from:
- namespaceSelector:
matchLabels:
app: web
podSelector:
matchLabels:
type: monitoring
k kubectl apply -f web-allow-all-ns-monitoring.yaml

B. kubectl create namespace web
kubectl label namespace/web app=web
vim web-allow-all-ns-monitoring.yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: web-allow-all-ns-monitoring
namespace: default
spec:
podSelector:
podSelector:
matchLabels:
type: monitoring
k kubectl apply -f web-allow-all-ns-monitoring.yaml

Answer: A

Question: 107

Create a ETCD backup of kubernetes cluster
Note : You don't need to memorize command, refer -
https://kubernetes.io/docs/tasks/administer-cluster/configureupgrade-etcd/ during exam

A. ETCDCTL_API=3 etcdctl --endpoints=[ENDPOINT] --cacert=[CA CERT]
--cert=[ETCD SERVER CERT] --key=[ETCD SERVER KEY] snapshot save
[BACKUP FILE NAME]
In exam, cluster setup is done with kubeadm , this means ETCD
used by the kubernetes cluster is coming from static pod.
kubectl get pod -n kube-system
kubectl describe pod etcd-master -n kube-system
You can locate the information on
endpoint: — advertise-client-urls=https://172.17.0.15:2379
ca certificate: — trusted-cafile=/etc/kubernetes/pki/etcd/ca.crt
server certificate : — certfile=/etc/kubernetes/pki/etcd/server.crt
key: — key-file=/etc/kubernetes/pki/etcd/server.key
To Create backup
export ETCDCTL_API=3
(or)
ETCDCTL_API=3 etcdctl ETCDCTL_API=3 etcdctl --
endpoints=https://172.17.0.15:2379 --
cacert=/etc/kubernetes/pki/etcd/ca.crt --
cert=/etc/kubernetes/pki/etcd/server.crt --
key=/etc/kubernetes/pki/etcd/server.key snapshot save etcdsnapshot.db
//Verify
ETCDCTL_API=3 etcdctl --write-out=table snapshot status
snapshot.db
B. ETCDCTL_API=3 etcdctl --endpoints=[ENDPOINT] --cacert=[CA CERT]
--cert=[ETCD SERVER CERT] --key=[ETCD SERVER KEY] snapshot save
[BACKUP FILE NAME]
In exam, cluster setup is done with kubeadm , this means ETCD
used by the kubernetes cluster is coming from static pod.
kubectl get pod -n kube-system
kubectl describe pod etcd-master -n kube-system
You can locate the information on
endpoint: — advertise-client-urls=https://172.16.0.18:2379
ca certificate: — trusted-cafile=/etc/kubernetes/pki/etcd/ca.crt
server certificate : — certfile=/etc/kubernetes/pki/etcd/server.crt
key: — key-file=/etc/kubernetes/pki/etcd/server.key
To Create backup
export ETCDCTL_API=3
(or)
ETCDCTL_API=3 etcdctl ETCDCTL_API=3 etcdctl --
endpoints=https://172.17.0.15:2379 --
key=/etc/kubernetes/pki/etcd/server.key snapshot save etcdsnapshot.db
//Verify
ETCDCTL_API=3 etcdctl --write-out=table snapshot status

snapshot.db

Answer: A

Question: 108

create a pod in a specific node (node1) by placing the pod definition file in a particular folder "/etc/kubernetes/manifests".

A. Generate YAML before we SSH to the specific node, then copy the YAML into the exam notepad to use it after SSH into worker node.
SSH to the node: "ssh node1"
Gain admin privileges to the node: "sudo -i"
Move to the manifest-path "cd /etc/kubernetes/manifests"
kubelet config file -- /var/lib/kubelet/config.yaml
Edit the config file "vi /var/lib/kubelet/config.yaml" to add staticPodPath
staticPodPath: /etc/kubernetes/manifests
Restart the kubelet "systemctl restart kubelet"

B. Generate YAML before we SSH to the specific node, then copy the YAML into the exam notepad to use it after SSH into worker node.
SSH to the node: "ssh node1"
Gain admin privileges to the node: "sudo -i"
Move to the manifest-path "cd /etc/kubernetes/manifests"
Place the generated YAML into the folder "vi nginx.yaml"
Find the kubelet config file path "ps -aux | grep kubelet" . This will output information on kubelet process. Locate the kubelet config file location.
kubelet config file -- /var/lib/kubelet/config.yaml
Edit the config file "vi /var/lib/kubelet/config.yaml" to add staticPodPath
staticPodPath: /etc/kubernetes/manifests
Restart the kubelet "systemctl restart kubelet"

Answer: B

Question: 109

CORRECT TEXT
View certificate details in /etc/kubernetes/pki

Answer:

// Verify Public Key / Certificate file
Openssl x509 -in certificate.crt -noout -text
// Verify Private Key
Openssl rsa -in "private-key" -check
// Verify CSR

Openssl req -text -noout -verify

Question: 110

CORRECT TEXT
Verify certificate expiry date for ca certificate in /etc/kubernetes/pki

Answer:

openssl x509 -in ca.crt -noout -text | grep -i validity -A 4

Question: 111

Evict all existing pods from a node-1 and make the node unschedulable for new pods.

A. kubectl get nodes
kubectl drain node-1 #It will evict pods running on node-1 to
other nodes in the cluster
kubectl cordon node-1 # New pods cannot be scheduled to the
node
// Verify
kubectl get no
When you cordon a node, the status shows “SchedulingDisabled”

B. kubectl get nodes
kubectl drain node-1 #It will evict pods running on node-1 to
other nodes in the cluster
// Verify
kubectl get no
When you cordon a node, the status shows “SchedulingDisabled”

Answer: A

Question: 112

CORRECT TEXT
Make the node schedulable by uncordon the node

Answer:

kubectl uncordon node-1
//verify
kubectl get no

Question: 113

Check nodes which are ready and print it to a file /opt/nodestatus

A. JSONPATH='{range .items[*]}{@.metadata.name}:{range
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
&& kubectl get nodes -o jsonpath="$JSONPATH" | grep
"Ready=True" > /opt/node-status
//Verify

cat /opt/node-status
B. JSONPATH='{range .items[*]}{@.metadata.name}:{range
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
//Verify
cat /opt/node-status

Answer: A

Question: 114

Install a kubernetes cluster with one master and one worker using kubeadm

A. This is a straightforward question, you need to install kubernetes cluster using kubeadm with one master and one worker.
Refer : https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/
B. This is a straightforward question, you need to install kubernetes cluster using kubeadm with one master and one worker.
Installation is considered success once both master and worker
nodes become available.
Refer : https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/

Answer: B

Question: 115

CORRECT TEXT
Get list of PVs and order by size and write to file ”/opt/pvstorage.txt”

Answer:

kubectl get pv --sort-by=.spec.capacity.storage > /opt/pv
storage.txt

Question: 116

Get the number of schedulable nodes and write to a file
/opt/schedulable-nodes.txt

A. kubectl get nodes -o jsonpath="{range
.items[*]}{.metadata.name}
{.spec.taints[?(@.effect=='NoSchedule')].effect}{\"\n\"}{end}"
| awk 'NF==1 {print $0}' > /opt/schedulable-nodes.txt
// Verify
cat /opt/schedulable-nodes.txt
B. kubectl get nodes -o jsonpath="{range
.items[*]}{.metadata.name}
{.spec.taints[?(@.effect=='NoSchedule')].effect}{\"\n\"}{end}"
| awk 'NF==11 {print $0}' > /opt/schedulable-nodes.txt

// Verify
cat /opt/schedulable-nodes.txt

Answer: A

Question: 117

Fix a node that shows as non-ready

A. Kubectl get nodes
// Check which node shows a not ready
kubectl describe nodes “node-name”
// Login to the node which shows as not ready and check the process for kubelet, docker , kube-proxy.
// systemctl status kubelet (or) ps -aux | grep -i “processname”
// If the process is not started, then start using
systemctl start kubelet / docker
// Verify
ps -auxxww | grep -i “process-name”
kubectl get nodes
B. Kubectl get nodes
// Check which node shows a not ready
kubectl describe nodes “node-name”
// Login to the node which shows as not ready and check the
systemctl start kubelet / docker
// Verify
ps -auxxww | grep -i “process-name”
kubectl get nodes

Answer: A

Question: 118

CORRECT TEXT
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

Question: 119

Deploy a pod with image=redis on a node with label disktype=ssd

A. // Get list of nodes

kubectl get nodes
//Get node with the label disktype=ssd
kubectl get no -l disktype=ssd
// Create a sample yaml file
kubectl run node-redis --generator=run-pod/v1 --image=redis --dry
run -o yaml > test-redis.yaml
// Edit test-redis.yaml file and add nodeSelector
vim test-redis.yaml
apiVersion: v1
kind: Pod
metadata:
name: redis
spec:
nodeSelector:
disktype: ssd
containers:
- name: node-redis
image: redis
imagePullPolicy: IfNotPresent
kubectl apply -f test-redis.yaml
/ // Verify
K kubectl get po -o wide

B. // Get list of nodes
kubectl get nodes
//Get node with the label disktype=ssd
kubectl get no -l disktype=ssd
// Create a sample yaml file
kubectl run node-redis --generator=run-pod/v1 --image=redis --dry
run -o yaml > test-redis.yaml
// Edit test-redis.yaml file and add nodeSelector
vim test-redis.yaml
apiVersion: v1
- name: node-redis
image: redis
imagePullPolicy: IfNotPresent
kubectl apply -f test-redis.yaml
/ // Verify
K kubectl get po -o wide

Answer: A

Question: 120

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
.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
// 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}'

Answer: B

Question: 121

CORRECT TEXT
Remove taint added to node “worker-2”

Answer:

kubectl taint nodes worker-2 key:NoSchedule-
// Verify
You will see a message “node/worker-2 untainted”
kubectl get nodes -o customcolumns=NAME:.metadata.name,TAINTS:.spec.taints --no-headers

Question: 122

CORRECT TEXT
Print all pod name and all image name and write it to a file
name "/opt/pod-details.txt"

Answer:

kubectl get pods -o=custom-columns='Pod
Name:metadata.name','Image:spec.containers[*].image' >
/opt/pod-details.txt

Thank You for Purchasing CKA PDF

Bonus Products

3 SETS OF REAL EXAM SCREENSHOTS - 2020

Starting from next page , there are 3 complete sets of real exam screenshots taken from testing center. It is a bonus product alongwith the PDF With questions and answers that you have read above this page. All you have to do is to find the correct answers to these real exam screenshots before attempting the real exam.

First SET starts below

BEFORE YOU BEGIN: IMPORTANT INSTRUCTIONS:

Each question on this exam must be completed on a designated *cluster*/configuration context. To minimize switching, the questions are grouped so that all questions on a given *cluster* appear consecutively.

There are six *clusters* that comprise the exam environment, made up of varying numbers of *containers*, as follows:

- **k8s** - 1 *master*, 2 *worker*
- **hk8s** - 1 *master*, 2 *worker*
- **bk8s** - 1 *master*, 1 *worker*
- **wk8s** - 1 *master*, 2 *worker*
- **ek8s** - 1 *master*, 2 *worker*
- **ik8s** - 1 *master*, 1 *base node*

At the start of each question you'll be provided with the command to ensure you are on the correct *cluster* e.g.

Set configuration context: `$ kubectl config use-context k8s`

nodes making up each *cluster* can be reached via `ssh`, using a command such as the following:

varying numbers of *containers*, as follows:

- **k8s** - 1 *master*, 2 *worker*
- **hk8s** - 1 *master*, 2 *worker*
- **bk8s** - 1 *master*, 1 *worker*
- **wk8s** - 1 *master*, 2 *worker*
- **ek8s** - 1 *master*, 2 *worker*
- **ik8s** - 1 *master*, 1 *base node*

At the start of each question you'll be provided with the command to ensure you are on the correct *cluster* e.g.

Set configuration context: `$ kubectl config use-context k8s`

nodes making up each *cluster* can be reached via `ssh`, using a command such as the following:

```
$ ssh k8s-node-0
```

Elevated privileges can be assumed on any *node* with the following command:

```
$ sudo -i
```

When you have finished working on a *node*, you should return to the base *node* (with hostname `node-1`) before attempting any further

```
$ ssh k8s-node-0
```

Elevated privileges can be assumed on any *node* with the following command:

```
$ sudo -i
```

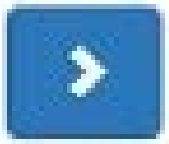
When you have finished working on a *node*, you should return to the *base node* (with hostname `node-1`) before attempting any further questions. Nested- `ssh` is not supported.

You can use `kubectl` and the appropriate context to work on any *cluster* from the *base node*. When connected to a *cluster* Member via `ssh`, you will only be able to work on that particular *cluster* via `kubectl`.

Further instructions for connecting to *cluster nodes* will be provided in the appropriate questions.

Please use the navigation panel to proceed to your first question.

Question: 0/24



Navigate All Questions

Question weight: 5%

Set configuration context: `$ kubectl config use-context k8s`

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

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

Question: 1/24

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

List all *persistent volumes* sorted by **name**, saving the full `kubectl` output to `/opt/KUCC00102/pv_list`. Use `kubectl`s own functionality for sorting the output, and do not manipulate it any further.

Question: 2/24

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

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

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

Question: 3/24



Navigate All Questions

Question weight: 7%

Set configuration context: `$ kubectl config use-context k8s`

Perform the following tasks:

- Add an *init container* to `lumpy-koala` (which has been defined in *spec* file `/opt/KUCC00108/pod-spec-KUCC00108.yaml`)
- The *init container* should create an empty file named `/workdir/eager.txt`
- If `/workdir/eager.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

Question: 4/24

Question 4

Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

Create a *pod* named `kucc4` 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` .

Question: 5/24



Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

Create a *pod* named `kucc4` 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` .

Question: 5/24



Navigate All Questions

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

Schedule a *pod* as follows:

- Name: `nginx-kusc00101`
- *Image*: `nginx`
- *Node selector*: `disk=spinning`

Question: 6/24

Question weight 4%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment* as follows:

- Name: `nginx-app`
- Using *container* `nginx` with version `1.10.2-alpine`
- The *deployment* should contain `3` replicas

Next, deploy the application with new version `1.13.0-alpine`, by performing a rolling update, and record that update.

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

Question: 7/24



Question 7

Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

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

Question: 8/24

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a *pod* as follows:

- Name: `nginx`
- Using *image*: `nginx`
- In a new *Kubernetes namespace* named: `website-frontend`

Question: 9/24

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment spec* file that will:

- Launch `7` replicas of the `redis` *image* with the *label* `app_runtime_stage=test`
- *deployment* name: `kual00201`

Save a copy of this *spec* file to `/opt/KUAL00201/spec_deploy.yaml` (or `.json`).

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

Question: 10/24



Navigate All Questions

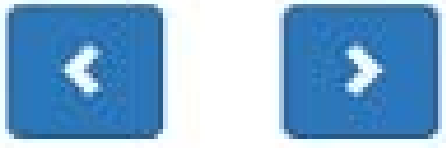
Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a file: `/opt/KUCC00302/kucc00302.txt` that lists all *pods* that implement *service* **foo** in *namespace* **production**.

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

Question: 11/24



Question weight: 9%

Set configuration context: `$ kubectl config use-context k8s`

Create a *Kubernetes secret* as follows:

- Name: `supersecret`
- username: `alice`

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 **username** as `TOPSECRET`.

Question: 12/24

Question 12

Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

Create a *pod* as follows:

- Name: `non-persistent-redis`
- *container image*: `redis`
- *Persistent volume* with name: `app-cache`
- Mount path: `/data/redis`

It should launch in the `staging` *namespace* and the *volume* **must not** be persistent.

Question: 13/24

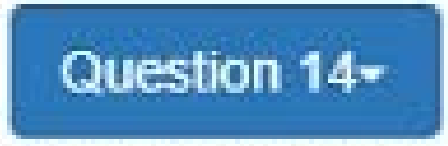
Question weight: 1%

Set configuration context: `$ kubectl config use-context k8s`

Scale the *deployment* **guestbook** to **3** *pods*.

Question: 14/24





Navigate All Questions

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

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

Question: 15/24

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

From the *pod label* **name=cpu-utilizer**, find *pods* running high CPU workloads and write the name of the *pod* consuming most CPU to the file `/opt/KUTR00102/KUTR00102.txt` (which already exists).

Question: 16/24

Question weight: 7%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment* as follows:

- Name: `nginx-test`
- Exposed via a *service* `nginx-test`
- 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.

Question: 17/24

‹ › Question 17▾
Navigate All Questions

Question weight: 7%

No configuration context change required for this item

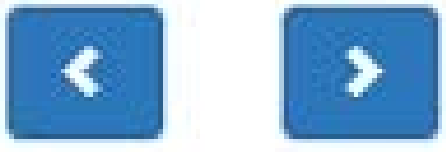
Create a snapshot of the `etcd` instance running at `https://127.0.0.1:2379`, saving the snapshot to the file path `/var/lib/backup/etcd-snapshot.db`.

The `etcd` instance is running `etcd` version 3.3.10.

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

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

Question: 18/24



Question 18

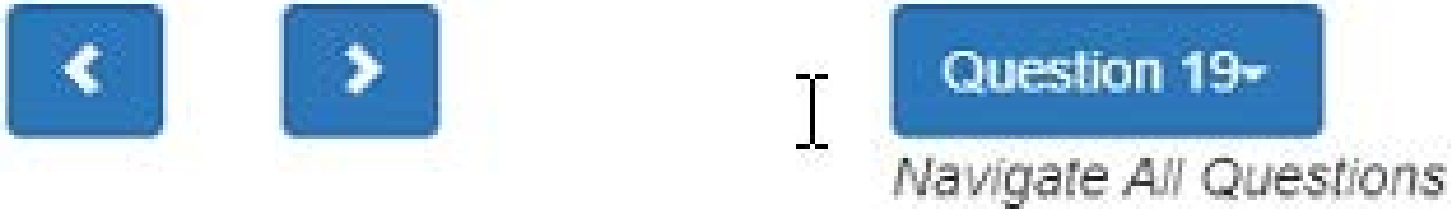
Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context ek8s`

Set the *node* named `ek8s-node-0` as unavailable and reschedule all the *pods* running on it.

Question: 19/24



Question weight: 4%

Set configuration context: `$ kubectl config use-context wk8s`

A *Kubernetes worker node*, named **wk8s-node-0** is in state `NotReady`. Investigate why this is the case, and perform any appropriate steps to bring the *node* to a `Ready` state, ensuring that any changes are made permanent.

Hints:

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

Question: 20/24



Navigate All Questions

Question weight: 4%

Set configuration context: `$ kubectl config use-context wk8s`

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

Hints:

- You can `ssh` to the appropriate *node* using: `$ ssh wk8s-node-1`
- You can assume elevated privileges on the *node* with the following command: `$ sudo -i`

Question: 21/24

Navigate All Questions

Question weight: 8%

No configuration context change required for this item

Important: For this item, you will have to `ssh` to the *nodes* `ik8s-master-0` and `ik8s-node-0` and complete all tasks on these *nodes*. Ensure that you return to the *base node* (hostname: `node-1`) when you have completed this item.

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*.

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

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

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

Ensure that you return to the *base node* (hostname: `node-1`) when you have completed this item.

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*.

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

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

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

The *cluster* will be considered complete once both *nodes* are in a **Ready** state.

Question: 22/24

Question weight: 4%

Set configuration context: `$ kubectl config use-context bk8s`

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.

Hints:

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

Question: 23/24

Question weight: 3%

Set configuration context: `$ kubectl config use-context hk8s`

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

Question: 24/24

Question 24

Navigate All Questions

2nd Set Starts Below

English

Question weight: 5%

Set configuration context: `$ kubectl config use-context k8s`

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

- Extract log lines corresponding to error `file-not-found`
- Write them to `/opt/KULM00201/foo`

Question: 1/24


Question 1

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

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.

Question: 2/24

Question 2

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

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

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

Question: 3/24

Question 3

Navigate All Questions

English

Question weight: 7%

Set configuration context: `$ kubectl config use-context k8s`

Perform the following tasks:

- Add an *init container* to `hungry-bear` (which has been defined in *spec* file `/opt/KUCC00108/pod-spec-KUCC00108.yaml`)
- The *init container* should create an empty file named `/workdir/eager.txt`
- If `/workdir/eager.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

Question: 4/24

Question 4

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

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`.

Question: 5/24


Question 5

Navigate All Questions

English

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

Schedule a *pod* as follows:

- Name: `nginx-kusc00101`
- *Image*: `nginx`
- *Node selector*: `disk=spinning`

Question: 6/24

Question 6

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment* as follows:

- Name: `nginx-app`
- Using *container* `nginx` with version `1.11.9-alpine`
- The *deployment* should contain `3` replicas

Next, deploy the application with new version `1.12.0-alpine`, by performing a rolling update, and record that update.

Finally, rollback that update to the previous version `1.11.9-alpine`.

Question: 7/24

Question 7

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

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

Question: 8/24


Question 8

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a *pod* as follows:

- Name: `mongo`
- Using *image*: `mongo`
- In a new *Kubernetes namespace* named: `website-backend`

Question: 9/24


Question 9

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment spec* file that will:

- Launch `3` replicas of the `nginx` *image* with the *label* `app_runtime_stage=dev`
- *deployment* name: `kual00201`

Save a copy of this *spec* file to `/opt/KUAL00201/deployment_spec.yaml` (or `.json`).

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

Question: 10/24

Question 10

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context k8s`

Create a file: `/opt/KUCC00302/kucc00302.txt` that lists all *pods* that implement *service* **baz** in *namespace* **development**.

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

Question: 11/24



Navigate All Questions

English

Question weight: 9%

Set configuration context: `$ kubectl config use-context k8s`

Create a *Kubernetes secret* as follows:

- Name: `super-secret`
- credential: `s3kr3t`

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

Create a second *pod* named `pod-secrets-via-env`, using the ***redis*** *image*, which exports **credential** as `CREDENTIALS`.

Question: 12/24

Question 12

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context k8s`

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 `pre-prod` *namespace* and the *volume* **must not** be persistent.

Question: 13/24


Question 13

Navigate All Questions

English

Question weight: 1%

Set configuration context: `$ kubectl config use-context k8s`

Scale the *deployment* **guestbook** to 3 *pods*.

Question: 14/24


Question 14

Navigate All Questions

English

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

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

Question: 15/24


Question 15

Navigate All Questions

English

Question weight: 2%

Set configuration context: `$ kubectl config use-context k8s`

From the *pod label* **name=overloaded-cpu**, 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).

Question: 16/24

Question 16

Navigate All Questions

English

Question weight: 7%

Set configuration context: `$ kubectl config use-context k8s`

Create a *deployment* as follows:

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

Next, use the utility `nslookup` to look up the *DNS* records of the *service* & *pod* and write the output to `/opt/KUNW00601/service.dns` and `/opt/KUNW00601/pod.dns` respectively. Ensure you use the `busybox:1.28` *image* (or earlier) for any testing, as the latest release has an *upstream* bug which impacts the use of `nslookup`.

Question: 17/24

Question 17

English

Question weight: 7%

No configuration context change required for this item

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

The `etcd` instance is running `etcd` version 3.3.10.

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

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

Question: 18/24


Question 18

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context ek8s`

Set the *node* named `ek8s-node-0` as unavailable and reschedule all the *pods* running on it.

Question: 19/24


Question 19

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context wk8s`

A *Kubernetes worker node*, named **wk8s-node-0** is in state `NotReady`. Investigate why this is the case, and perform any appropriate steps to bring the *node* to a `Ready` state, ensuring that any changes are made permanent.

Hints:

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

Question: 20/24


Question 20

Navigate All Questions

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context wk8s`

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

Hints:

- You can `ssh` to the appropriate *node* using: `$ ssh wk8s-node-1`
- You can assume elevated privileges on the *node* with the following command: `$ sudo -i`

Question: 21/24


Question 21

Navigate All Questions

No configuration context change required for this item

Important: For this item, you will have to `ssh` to the *nodes* `ik8s-master-0` and `ik8s-node-0` and complete all tasks on these *nodes*. Ensure that you return to the *base node* (hostname: `node-1`) when you have completed this item.

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*.

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

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

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

The *cluster* will be considered complete once both *nodes* are in a **Ready** state.

Question: 22/24

English

Question weight: 4%

Set configuration context: `$ kubectl config use-context bk8s`

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.

Hints:

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

Question: 23/24

Question 23

Navigate All Questions

English

Question weight: 3%

Set configuration context: `$ kubectl config use-context hk8s`

Create a *persistent volume* with name `app-data`, of capacity `1Gi` and access mode `ReadWriteMany`. The type of *volume* is `hostPath` and its location is `/srv/app-data`.

Question: 24/24

Question 24

Navigate All Questions

3rd Set Starts Below

Task weight: 5%

Set configuration context:

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

Task

Monitor the logs of pod `bar` and:

- Extract log lines corresponding to error `file-not-found`
- Write them to `/opt/KULM00201/bar`

Task weight: 5%

Set configuration context:

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

Task

Monitor the logs of pod bar and:

- Extract log lines corresponding to error file-not-found
- Write them to /opt/KULM00201/bar

Flag this to ...

I am satisfied, ...

Task weight: 3%

Set configuration context:

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

Task

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

Flag this to ...

I am satisfied, ...

Task weight: 7%

Set configuration context:

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

Task

Perform the following tasks:

- Add an init container to `lumpy-koala` (which has been defined in spec file `/opt/KUCC00108/pod-spec-KUCC00108.yaml`)
- The init container should create an empty file named `/workdir/faithful.txt`
- If `/workdir/faithful.txt` is not

Perform the following tasks:

- Add an init container to lumpy-koala (which has been defined in spec file /opt/KUCC00108/pod-spec-KUCC00108.yaml)
- The init container should create an empty file named /workdir/faithful.txt
- If /workdir/faithful.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

Flag this to ... | I am satisfied, ...

Task weight: 3%

Set configuration context:

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

Task

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

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

Task weight: 4%

Set configuration context:

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

Task

Create a pod named kucc1 with a single app container for each of the following images running inside (there may be between 1 and 4 images specified): nginx + redis.

Flag this to ...

I am satisfied, ...

Task weight: 2%

Set configuration context:

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

Task

Schedule a pod as follows:

- Name: nginx-kusc00101
- Image: nginx
- Node selector: disk=spinning

Flag this to ...

I am satisfied, ...

Task weight: 4%

Set configuration context:

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

Task

Create a deployment as follows:

- Name: nginx-app
- Using container nginx with version 1.11.9-alpine
- The deployment should contain 3 replicas

Next, deploy the application with new version 1.13.0-alpine , by performing a rolling update.

Finally, rollback that update to the

```
config use-context k8s
```

Task

Create a deployment as follows:

- Name: nginx-app
- Using container nginx with version 1.11.9-alpine
- The deployment should contain 3 replicas

Next, deploy the application with new version 1.13.0-alpine , by performing a rolling update.

Finally, rollback that update to the previous version 1.11.9-alpine .

Flag this to ...

I am satisfied, ...

Task weight: 4%

Set configuration context:

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

Task

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

Flag this to ...

I am satisfied, ...

Task weight: 3%

Set configuration context:

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

Task

Create a pod as follows:

- Name: nginx
- Using image: nginx
- In a new Kubernetes namespace named: website-backend

Task weight: 3%

Set configuration context:

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

Task

Create a deployment spec file that will:

- Launch 4 replicas of the `redis` image with the label `app_runtime_mode=prod`
- deployment name: `kual00201`

Save a copy of this spec file to `/opt/KUAL00201/deploy_spec.yaml` (or `/opt/KUAL00201/deploy_spec.json`).

When you are done, clean up (delete)

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

Task

Create a deployment spec file that will:

- Launch 4 replicas of the redis image with the label app_runtime_mode=prod
- deployment name: kual00201

Save a copy of this spec file to /opt/KUAL00201/deploy_spec.yaml (or /opt/KUAL00201/deploy_spec.json).

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

Flag this to ... | I am satisfied, ...

Task weight: 3%

Set configuration context:

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

Task

Create a file: `/opt/KUCC00302/kucc00302.txt` that lists all pods that implement service `bar` in namespace `development`.

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

Flag this to ...

I am satisfied, ...

Set configuration context:

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

Task

Create a Kubernetes secret as follows:

- Name: super-secret
- password : s3kr3t

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 TOPSECRET .

Task weight: 4%

Set configuration context:

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

Task

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 pre-prod namespace and the volume **must not** be persistent.

Task weight: 1%

Set configuration context:

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

Task

Scale the deployment loadbalancer to 3 pods.

Flag this to ...

I am satisfied, ...

Task weight: 2%

Set configuration context:

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

Task

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

Task weight: 2%

Set configuration context:

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

Task

From the pod label `name=overloaded-cpu`, 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).

Task weight: 7%

Set configuration context:

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

Task

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

Task

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.

Task weight: 7%

No configuration context change required for this item.

Task

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

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

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

saving the snapshot to the file path /data/backup/etcd-snapshot.db .

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

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

Flag this to ... | I am satisfied, ...

Task weight: 4%

Set configuration context:

```
[student@node-1] $ kubectl config use-context ek8s
```

Task

Set the node named `ek8s-node-1` as unavailable and reschedule all the pods running on it.

Flag this to ...

I am satisfied, ...

Task weight: 4%

Set configuration context:

```
[student@node-1] $ kubectl config use-context wk8s
```

Task

A Kubernetes worker node, named `wk8s-node-0` is in state `NotReady`. Investigate why this is the case, and perform any appropriate steps to bring the node to a `Ready` state, ensuring that any changes are made permanent.

You can `ssh` to the failed node using:

```
[student@node-1] $ ssh wk8
```

perform any appropriate steps to bring the node to a `Ready` state, ensuring that any changes are made permanent.

You can `ssh` to the failed node using:

```
[student@node-1] $ ssh wk8s-node-0
```

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

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


Flag this to ... | I am satisfied, ...

Task weight: 4%

Set configuration context:

```
[student@node-1] $ kubectl config use-context wk8s
```

Task

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 `jenkins` named `app` 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:

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
```


Flag this to ... | I am satisfied, ...

Task weight: 8%

No configuration context change required for this item.

For this item, you will have to ssh to the nodes ik8s-master-0 and ik8s-node-0 and complete all tasks on these nodes. Ensure that you return to the base node (hostname: node-1) when you have completed this item.

Context

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

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

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

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

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

The cluster will be considered complete once both nodes are in a **Ready** state.

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

Flag this to ... | I am satisfied, ...

Task weight: 4%

Set configuration context:

```
[student@node-1] $ kubectl config use-context bk8s
```

Task

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

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 nodes (`bk8s-master-0` or `bk8s-node-0`) using:

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

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

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

made permanently.

You can ssh to the relevant nodes (bk8s-master-0 or bk8s-node-0) using:

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

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

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


Flag this to ...

I am satisfied, ...

Task weight: 3%

Set configuration context:

```
[student@node-1] $ kubectl config use-context hk8s
```

Task

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

Flag this to ...