Que:

---

**Run a pod called alpine-sleeper-cka15-arch using the alpine image in the default namespace that will sleep for 7200 seconds.**

Ans:

---

Create the pod definition:

student-node ~ ➜ vi alpine-sleeper-cka15-arch.yaml

##Content should be:

apiVersion: v1

kind: Pod

metadata:

name: alpine-sleeper-cka15-arch

spec:

containers:

- name: alpine

image: alpine

command: ["/bin/sh", "-c", "sleep 7200"]

================================================

Que:

---

**We have created a service account called red-sa-cka23-arch, a cluster role called red-role-cka23-arch and a cluster role binding called red-role-binding-cka23-arch.**

**Identify the permissions of this service account and write down the answer in file /opt/red-sa-cka23-arch in format resource:pods|verbs:get,list on student-node**

Ans:

---

Get the red-role-cka23-arch role permissions:

student-node ~ ➜ kubectl get clusterrole red-role-cka23-arch -o json --context cluster1

{

"apiVersion": "rbac.authorization.k8s.io/v1",

"kind": "ClusterRole",

"metadata": {

"creationTimestamp": "2022-10-20T07:16:39Z",

"name": "red-role-cka23-arch",

"resourceVersion": "16324",

"uid": "e53cef4f-ae1b-49f7-b9fa-ac5e7e22a61c"

},

"rules": [

{

"apiGroups": [

"apps"

],

"resources": [

"deployments"

],

"verbs": [

"get",

"list",

"watch"

]

}

]

}

In this case, add data in file as below:

student-node ~ ➜ echo "resource:deployments|verbs:get,list,watch" > /opt/red-sa-cka23-arch

=====================================================

**Que:**

**---**

**There is a Cronjob called orange-cron-cka10-trb which is supposed to run every two minutes (i.e 13:02, 13:04, 13:06…14:02, 14:04…and so on). This cron targets the application running inside the orange-app-cka10-trb pod to make sure the app is accessible. The application has been exposed internally as a ClusterIP service.**

However, this cron is not running as per the expected schedule and is not running as intended.

Make the appropriate changes so that the cronjob runs as per the required schedule and it passes the accessibility checks every-time.

Ans:

---

Check the cron schedule

kubectl get cronjob

Make sure the schedule for orange-cron-cka10-trb crontjob is set to \*/2 \* \* \* \* if not then edit it.

Also before that look for the issues why this cron is failing

kubectl logs orange-cron-cka10-trb-xxxx

You will see some error like

curl: (6) Could not resolve host: orange-app-cka10-trb

You will notice that the curl is trying to hit orange-app-cka10-trb directly but it is supposed to hit the relevant service which is orange-svc-cka10-trb so we need to fix the curl command.

Edit the cronjob

kubectl edit cronjob orange-cron-cka10-trb

Change schedule \* \* \* \* \* to \*/2 \* \* \* \*

Change command curl orange-app-cka10-trb to curl orange-svc-cka10-trb

Wait for 2 minutes to run again this cron and it should complete now.

=========================================================

**Que:**

**---**

**The blue-dp-cka09-trb deployment is having 0 out of 1 pods running. Fix the issue to make sure that pod is up and running.**

Ans:

---

List the pods

kubectl get pod

Most probably you see Init:Error or Init:CrashLoopBackOff for the corresponding pod.

Look into the logs

kubectl logs blue-dp-cka09-trb-xxxx -c init-container

You will see an error something like

sh: can't open 'echo 'Welcome!'': No such file or directory

Edit the deployment

kubectl edit deploy blue-dp-cka09-trb

Under initContainers: -> - command: add -c to the next line of - sh, so final command should look like this

initContainers:

- command:

- sh

- -c

- echo 'Welcome!'

If you will check pod then it must be failing again but with different error this time, let's find that out

kubectl get event --field-selector involvedObject.name=blue-dp-cka09-trb-xxxxx

You will see an error something like

Warning Failed pod/blue-dp-cka09-trb-69dd844f76-rv9z8 Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error mounting "/var/lib/kubelet/pods/98182a41-6d6d-406a-a3e2-37c33036acac/volumes/kubernetes.io~configmap/nginx-config" to rootfs at "/etc/nginx/nginx.conf": mount /var/lib/kubelet/pods/98182a41-6d6d-406a-a3e2-37c33036acac/volumes/kubernetes.io~configmap/nginx-config:/etc/nginx/nginx.conf (via /proc/self/fd/6), flags: 0x5001: not a directory: unknown

Edit the deployment again

kubectl edit deploy blue-dp-cka09-trb

Under volumeMounts: -> - mountPath: /etc/nginx/nginx.conf -> name: nginx-config add subPath: nginx.conf and save the changes.

Finally the pod should be in running state.

======================================================

**Que:**

**---**

**The deployment called web-dp-cka17-trb has 0 out of 1 pods up and running. Troubleshoot this issue and fix it. Make sure all required POD(s) are in running state and stable (not restarting).**

**The application runs on port 80 inside the container and is exposed on the node port 30090.**

Ans:

---

List out the PODs

kubectl get pod

Let's look into the relevant events:

kubectl get event --field-selector involvedObject.name=<pod-name>

You should see some errors as below:

Warning FailedScheduling pod/web-dp-cka17-trb-9bdd6779-fm95t 0/3 nodes are available: 3 persistentvolumeclaim "web-pvc-cka17-trbl" not found. preemption: 0/3 nodes are available: 3 Preemption is not helpful for scheduling.

From the error we can see that its something related to the PVCs. So let' look into that.

kubectl get pv

kubectl get pvc

You will notice that web-pvc-cka17-trb is stuck in pending and also the capacity of web-pv-cka17-trb volume is 100Mi.

Now let's dig more into the PVC:

kubectl get pvc web-pvc-cka17-trb -o yaml

Notice the storage which is 150Mi which means its trying to claim 150Mi of storage from a 100Mi PV. So let's edit this PV.

kubectl edit pv web-pv-cka17-trb

Change storage: 100Mi to storage: 150Mi

Check again the pvc

kubectl get pvc

web-pvc-cka17-trb should be good now. let's see the PODs

kubectl get pod

POD should not be in pending state now but it must be crashing with Init:CrashLoopBackOff status, which means somehow the init container is crashing. So let's check the logs.

kubectl get event --field-selector involvedObject.name=<pod-name>

You should see someting like

Warning Failed pod/web-dp-cka17-trb-67c9bdcd85-4tvpr Error: failed to create containerd task: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: "/bin/bsh\\": stat /bin/bsh\: no such file or directory: unknown

Let's look into the deployment:

kubectl edit deploy web-dp-cka17-trb

Under initContainers: -> - command: change /bin/bsh\ to /bin/bash

let's see the PODs

kubectl get pod

Wait for some time to make sure it is stable, but you will notice that its restart so still something must be wrong.

So let's check the events again.

kubectl get event --field-selector involvedObject.name=<pod-name>

You should see someting like

Warning Unhealthy pod/web-dp-cka17-trb-647f69f8bd-67xmx Liveness probe failed: Get "http://10.50.64.1:81/": dial tcp 10.50.64.1:81: connect: connection refused

Seems like its not able to connect to a service, let's look into the deployment to understand

kubectl edit deploy web-dp-cka17-trb

Notice that containerPort: 80 but under livenessProbe: the port: 81 so seems like livenessProbe is using wrong port. let's change port: 81 to port: 80

See the PODs now

kubectl get pod

It should be good now.

============================================================

**Que:**

**---**

**We tried to schedule grey-cka21-trb pod on cluster4 which was supposed to be deployed by the kubernetes scheduler so far but somehow its stuck in Pending state. Look into the issue and fix the same, make sure the pod is in Running state.**

You can SSH into the cluster4 using ssh cluster4-controlplane command.

Ans:

---

Follow below given steps

Let's check the POD status

kubectl get pod --context=cluster4

You will see that grey-cka21-trb pod is stuck in Pending state. So let's try to look into the logs and events

kubectl logs grey-cka21-trb --context=cluster4

kubectl get event --context=cluster4 --field-selector involvedObject.name=grey-cka21-trb

You might not find any relevant info in the logs/events. Let's check the status of the kube-scheduler pod

kubectl get pod --context=cluster4 -n kube-system

You will notice that kube-scheduler-cluster4-controlplane pod is crashing, let's look into its logs

kubectl logs kube-scheduler-cluster4-controlplane --context=cluster4 -n kube-system

You will see an error as below:

run.go:74] "command failed" err="failed to get delegated authentication kubeconfig: failed to get delegated authentication kubeconfig: stat /etc/kubernetes/scheduler.config: no such file or directory"

From the logs we can see that its looking for a file called /etc/kubernetes/scheduler.config which seems incorrect, let's look into the kube-scheduler manifest on cluster4.

ssh cluster4-controlplane

First let's find out if /etc/kubernetes/scheduler.config

ls /etc/kubernetes/scheduler.config

You won't find it, instead the correct file is /etc/kubernetes/scheduler.conf so let's modify the manifest.

vi /etc/kubernetes/manifests/kube-scheduler.yaml

Search for config in the file, you will find some typos, change every occurence of /etc/kubernetes/scheduler.config to /etc/kubernetes/scheduler.conf.

Let's see if kube-scheduler-cluster4-controlplane is running now

kubectl get pod -A

It should be good now and grey-cka21-trb should be good as well.

=================================================================

**Que:**

**---**

**The cat-cka22-trb pod is stuck in Pending state. Look into the issue to fix the same. Make sure that the pod is in running state and its stable (i.e not restarting or crashing).**

Note: Do not make any changes to the pod (No changes to pod config but you may destory and re-create).

Ans:

---

Let's check the POD status

kubectl get pod

You will see that cat-cka22-trb pod is stuck in Pending state. So let's try to look into the events

kubectl --context cluster2 get event --field-selector involvedObject.name=cat-cka22-trb

You will see some logs as below

Warning FailedScheduling pod/cat-cka22-trb 0/3 nodes are available: 1 node(s) had untolerated taint {node-role.kubernetes.io/master: }, 2 node(s) didn't match Pod's node affinity/selector. preemption: 0/2 nodes are available: 3 Preemption is not helpful for scheduling.

So seems like this POD is using the node affinity, let's look into the POD to understand the node affinity its using.

kubectl --context cluster2 get pod cat-cka22-trb -o yaml

Under affinity: you will see its looking for key: node and values: cluster2-node02 so let's verify if node01 has these labels applied.

kubectl --context cluster2 get node cluster2-node01 -o yaml

Look under labels: and you will not find any such label, so let's add this label to this node.

kubectl label node cluster1-node01 node=cluster2-node01

Check again the node details

kubectl get node cluster2-node01 -o yaml

The new label should be there, let's see if POD is scheduled now on this node

kubectl --context cluster2 get pod

Its is but it must be crashing or restarting, so let's look into the pod logs

kubectl --context cluster2 logs -f cat-cka22-trb

You will see logs as below:

The HOST variable seems incorrect, it must be set to kodekloud

Let's look into the POD env variables to see if there is any HOST env variable

kubectl --context cluster2 get pod -o yaml

Under env: you will see this

env:

- name: HOST

valueFrom:

secretKeyRef:

key: hostname

name: cat-cka22-trb

So we can see that HOST variable is defined and its value is being retrieved from a secret called "cat-cka22-trb". Let's look into this secret.

kubectl --context cluster2 get secret

kubectl --context cluster2 get secret cat-cka22-trb -o yaml

You will find a key/value pair under data:, let's try to decode it to see its value:

echo "<the decoded value you see for hostname" | base64 -d

ok so the value is set to kodekloude which is incorrect as it should be set to kodekloud. So let's update the secret:

echo "kodekloud" | base64

kubectl edit secret cat-cka22-trb

Change requests storage hostname: a29kZWtsb3Vkdg== to hostname: a29kZWtsb3VkCg== (values may vary)

POD should be good now.

=================================================

**Que:**

**---**

**Create a deployment called app-wl01 using the nginx image and scale the application pods to 2**

Ans:

---

Run the command to change the context: -

kubectl config use-context cluster1

Run the following commands: -

kubectl create deployment app-wl01 --image=nginx --replicas=2

To cross-verify the deployed resources, run the commands as follows: -

kubectl get pods,deployments

================================================

Que:

---

**We want to deploy a python based application on the cluster using a template located at /root/olive-app-cka10-str.yaml on student-node. However, before you proceed we need to make some modifications to the YAML file as per details given below:**

**The YAML should also contain a persistent volume claim with name olive-pvc-cka10-str to claim a 100Mi of storage from olive-pv-cka10-str PV.**

**Update the deployment to add a sidecar container, which can use busybox image (you might need to add a sleep command for this container to keep it running.)**

**Share the python-data volume with this container and mount the same at path /usr/src. Make sure this container only has read permissions on this volume.**

Finally, create a pod using this YAML and make sure the POD is in Running state.

Ans:

---

Update olive-app-cka10-str.yaml template so that it looks like as below:

---

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: olive-pvc-cka10-str

spec:

accessModes:

- ReadWriteMany

storageClassName: olive-stc-cka10-str

volumeName: olive-pv-cka10-str

resources:

requests:

storage: 100Mi

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: olive-app-cka10-str

spec:

replicas: 1

template:

metadata:

labels:

app: olive-app-cka10-str

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: kubernetes.io/hostname

operator: In

values:

- cluster1-node01

containers:

- name: python

image: poroko/flask-demo-app

ports:

- containerPort: 5000

volumeMounts:

- name: python-data

mountPath: /usr/share/

- name: busybox

image: busybox

command:

- "bin/sh"

- "-c"

- "sleep 10000"

volumeMounts:

- name: python-data

mountPath: "/usr/src"

readOnly: true

volumes:

- name: python-data

persistentVolumeClaim:

claimName: olive-pvc-cka10-str

selector:

matchLabels:

app: olive-app-cka10-str

---

apiVersion: v1

kind: Service

metadata:

name: olive-svc-cka10-str

spec:

type: NodePort

ports:

- port: 5000

nodePort: 32006

selector:

app: olive-app-cka10-str

Apply the template:

kubectl apply -f olive-app-cka10-str.yaml

====================================================

Que:

---

Create a nginx pod called nginx-resolver-cka06-svcn using image nginx, expose it internally with a service called nginx-resolver-service-cka06-svcn.

Test that you are able to look up the service and pod names from within the cluster. Use the image: busybox:1.28 for dns lookup. Record results in /root/CKA/nginx.svc.cka06.svcn and /root/CKA/nginx.pod.cka06.svcn

Ans:

---

To create a pod nginx-resolver-cka06-svcn and expose it internally:

student-node ~ ➜ kubectl run nginx-resolver-cka06-svcn --image=nginx

student-node ~ ➜ kubectl expose pod/nginx-resolver-cka06-svcn --name=nginx-resolver-service-cka06-svcn --port=80 --target-port=80 --type=ClusterIP

To create a pod test-nslookup. Test that you are able to look up the service and pod names from within the cluster:

student-node ~ ➜ kubectl run test-nslookup --image=busybox:1.28 --rm -it --restart=Never -- nslookup nginx-resolver-service-cka06-svcn

student-node ~ ➜ kubectl run test-nslookup --image=busybox:1.28 --rm -it --restart=Never -- nslookup nginx-resolver-service-cka06-svcn > /root/CKA/nginx.svc.cka06.svcn

Get the IP of the nginx-resolver-cka06-svcn pod and replace the dots(.) with hyphon(-) which will be used below.

student-node ~ ➜ kubectl get pod nginx-resolver-cka06-svcn -o wide

student-node ~ ➜ IP=`kubectl get pod nginx-resolver-cka06-svcn -o wide --no-headers | awk '{print $6}' | tr '.' '-'`

student-node ~ ➜ kubectl run test-nslookup --image=busybox:1.28 --rm -it --restart=Never -- nslookup $IP.default.pod > /root/CKA/nginx.pod.cka06.svcn

==================================================

Que:

---

We have an external webserver running on student-node which is exposed at port 9999. We have created a service called external-webserver-cka03-svcn that can connect to our local webserver from within the kubernetes cluster3 but at the moment it is not working as expected.

Fix the issue so that other pods within cluster3 can use external-webserver-cka03-svcn service to access the webserver.

Ans:

---

Let's check if the webserver is working or not:

student-node ~ ➜ curl student-node:9999

...

<h1>Welcome to nginx!</h1>

...

Now we will check if service is correctly defined:

student-node ~ ➜ kubectl describe svc external-webserver-cka03-svcn

Name: external-webserver-cka03-svcn

Namespace: default

.

.

Endpoints: <none> # there are no endpoints for the service

...

As we can see there is no endpoints specified for the service, hence we won't be able to get any output. Since we can not destroy any k8s object, let's create the endpoint manually for this service as shown below:

student-node ~ ➜ export IP\_ADDR=$(ifconfig eth0 | grep inet | awk '{print $2}')

student-node ~ ➜ kubectl --context cluster3 apply -f - <<EOF

apiVersion: v1

kind: Endpoints

metadata:

# the name here should match the name of the Service

name: external-webserver-cka03-svcn

subsets:

- addresses:

- ip: $IP\_ADDR

ports:

- port: 9999

EOF

Finally check if the curl test works now:

student-node ~ ➜ kubectl --context cluster3 run --rm -i test-curl-pod --image=curlimages/curl --restart=Never -- curl -m 2 external-webserver-cka03-svcn

...

<title>Welcome to nginx!</title>

...

===========================================================