# Introduction

Welcome to the Ultimate Certified Kubernetes administrator solutions document. This document comprises of ~ 55 unique curated CKA questions divided into different subcategories. The sole proprietor of these questions is [KodeKloud](https://kodekloud.com/) team. Ultimate CKA course is part of KodeKloud which had real time questions simulated randomly in a lab environment with a time limit of 2 hours. This course is a mimic of a real time CKA exam. Recently, I had an opportunity to work on this real time mock exam and would like to present the solutions for each unique questions as a document for the aspirants who are trying to clear CKA exams.

Going through this document will give an idea on how an issue/task should be approached from the scratch by understanding the question. Screen shots are provided under each question for better understanding. Please ensure you meet the following pre-requisites to continue with this further.

# Pre-requisite:

The mandatory pre-requisite to understand the upcoming question is to finish the below listed courses with practical exams.

* [Kubernetes for absolute beginners](https://kodekloud.com/courses/kubernetes-for-the-absolute-beginners-hands-on/)
* [Certified Kubernetes administrator with practise test](https://kodekloud.com/courses/labs-certified-kubernetes-administrator-with-practice-tests/)

Additional courses

* [Kubernetes challenges](https://kodekloud.com/courses/kubernetes-challenges/)

If you are interested to work on the Ultimate CKA labs directly, please take it by clicking on the [link](https://kodekloud.com/courses/ultimate-certified-kubernetes-administrator-cka-mock-exam/)

# Architecture, Install and Maintenance

**Question 1:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Create a generic secret called secure-sec-cka12-arch in the secure-sys-cka12-arch namespace on the cluster3. Use the key/value of color=darkblue to create the secret.

**Solution:**

The question is straight forward to create a secret in a namespace. So lets create a secret on the namespace with the values as follows.

student-node ~ ➜ kubectl create secret generic secure-sec-cka12-arch -n secure-sys-cka12-arch --from-literal=color=darkblue

secret/secure-sec-cka12-arch created

The secret has been validated using the below commands.

student-node ~ ➜ k get secrets -n secure-sys-cka12-arch

NAME TYPE DATA AGE

secure-sec-cka12-arch Opaque 1 7s

student-node ~ ➜ k describe secrets -n secure-sys-cka12-arch secure-sec-cka12-arch

Name: secure-sec-cka12-arch

Namespace: secure-sys-cka12-arch

Labels: <none>

Annotations: <none>

Type: Opaque

Data

====

color: 8 bytes

**Question 2:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

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

* Correct resources added?
* Correct verbs used?

**Solution:**

Upon switching the context to cluster1, let us describe the cluster role to understand the permissions set on the service account. From the below screen shot, its clear that the service account has get list and watch permissions on the resource “deployments”. So let us write that into the destinated file and save it.

The same file has been validated by opening the file.

Text

Description automatically generated

**Question 3:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

A pod called logger-complete-cka04-arch has been created in the default namespace. Inspect this pod and save ALL the logs to the file /root/logger-complete-cka04-arch on the student-node.

* Task completed?

**Solution:**

* As a first step, switch to cluster 3 using the given command in the first section.
* View the logs of the pod by supplying “kubectl logs <pod\_name>” and redirect the logs to the given file name
* Validate the same by opening the file using cat command.

Text

Description automatically generated

**Question 4:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

ETCD Key Value store which is running as a pod in cluster1. Take the backup of it and store it on the cluster1-controlplane node at the path /opt/cluster1\_backup.db.

You can ssh to the controlplane node by running ssh root@cluster1-controlplane from the student-node.

NOTE: - If the etcd utility tool is unavailable on the controlplane, install it first.

* etcd backup complete?

**Solution:**

* Switch to the cluster1 and login to control plane node.
* As etcd clusters are deployed as static pods on the master node. View the yaml files in the desired path /etc/Kubernetes/manifests/etcd.yaml.

Text

Description automatically generated

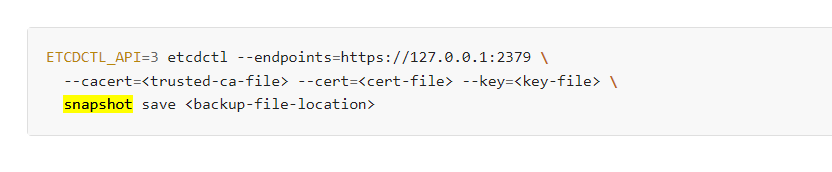
* Open the file and copy the necessary values in a note pad.
  + Etcd endpoint.
  + Server certificate.
  + CA certificate.
  + Key file.

A black screen with white text

Description automatically generated with low confidence

Click on the official documentation provided in the questionnaire to refer the command to take the etcd backup.

Supply the vlaues into the command and execute it to save the backup in the desired location.



ETCDCTL\_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \

--cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \

snapshot save /opt/cluster1\_backup.db

This can be validated by checking the snapshot status command given in the official documentation.

A screenshot of a computer

Description automatically generated

**Question 5:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Decode the existing secret called beta-sec-cka14-arch created in the beta-ns-cka14-arch namespace and store the decoded content inside the file /opt/beta-sec-cka14-arch on the student-node.

* secret decoded?

**Solution:**

* Switch the context to cluster 3.
* As per your convenience, switch the default namespace to the given namespace (You could directly supply that in the command without switching the namespace as well)
* Describe and edit the secret to get the value of the secret.

Text

Description automatically generated

Follow the instructions, in the official documentation to decode the secret name.

Link: <https://kubernetes.io/docs/tasks/configmap-secret/managing-secret-using-kubectl/>

As shown in the screen shot the secret has been decoded successfully. Now its time to save the same to the given file name.

Text

Description automatically generated

**Question 6:**

Find the node across all clusters that consumes the most CPU and store the result to the file /opt/high\_cpu\_node in the following format cluster\_name,node\_name.

The node could be in any clusters that are currently configured on the student-node.

* data stored in /opt/high\_cpu\_node?

**Solution:**

The ask is to identify the nodes which consume the highest CPU across all the clusters. So let us understand the number of clusters we have in total.

Text

Description automatically generated

Then switch into each cluster and execute the command to list down the nodes which consume most CPU.

k config use-context cluster1

K top nodes

k config use-context cluster2

K top nodes

k config use-context cluster3

K top nodes

k config use-context cluster4

K top nodes

Text

Description automatically generated

From the output, we understand the cluster 4 controlplan node had consumed the highest CPU, so lets write that information into the destinated file.

**Question 7:**

Find the pod that consumes the most memory and store the result to the file /opt/high\_memory\_pod in the following format cluster\_name,namespace,pod\_name.

The pod could be in any namespace in any of the clusters that are currently configured on the student-node.

* data stored in /opt/high\_memory\_pod?

**Solution:**

The solution is pretty straight forward as we did in the previous question.

Text

Description automatically generated

k config use-context cluster1

K top pods -A

k config use-context cluster2

K top pods -A

k config use-context cluster3

K top pods -A

k config use-context cluster4

K top pods -A

The high consumed memory had been written onto the destinated file.

Text

Description automatically generated

**Question 8:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Run a pod called looper-cka16-arch using the busybox image that runs the while loop while true; do echo hello; sleep 10;done. This pod should be created in the default namespace.

* looper-cka16-arch pod created?
* pod prints hello every 10 seconds?

**Solution:**

First, lets switch to the cluster and create an imperative command to create the yaml file and save it in a temporary file.

Text

Description automatically generated

Now, lets edit the file and supply the command in the yaml file and save it.

Text

Description automatically generated

We can validate this by listing the pods and the logs which prints “hello” successfully.

Text

Description automatically generated

**Question 9:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

Create a service account called deploy-cka20-arch. Further create a cluster role called deploy-role-cka20-arch with permissions to get the deployments in default namespace on cluster1.

Finally create a cluster role binding called deploy-role-binding-cka20-arch to bind deploy-role-cka20-arch cluster role with deploy-cka20-arch service account.

* Task completed?

**Solution:**

The solution approach follows the below sequence.

* Switch to the desired cluster.
* Create a service account.
* Create a cluster role.
* Create a cluster role binding to bind the role to the service account.

From the screen shot, we could see the SA created and the cluster role imperative command has been supplied to a temporary file.

Text

Description automatically generated

Cluster role edited to update the appropriate permissions.

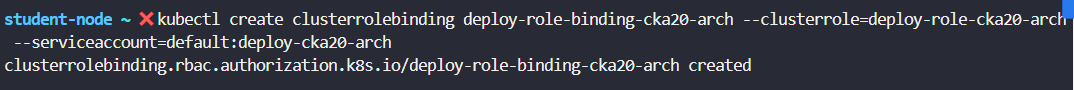
Text

Description automatically generated

Text

Description automatically generated

Cluster role binding created.



To validate this, describe the clusterrole and view the permissions.

Text

Description automatically generated

Text

Description automatically generated

In order to validate if the service account can get the deployments, below imperative command is used.

Note: Service account name in the “--as” field should be called out with a specific format “system:serviceaccount:<*Namespace\_Name>:<Serviceaccountname>*”.

For more details, refer the official documentation.



**Question 10:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

A pod called elastic-app-cka02-arch is running in the default namespace. The YAML file for this pod is available at /root/elastic-app-cka02-arch.yaml on the student-node. The single application container in this pod writes logs to the file /var/log/elastic-app.log.

One of our logging mechanisms needs to read these logs to send them to an upstream logging server but we don't want to increase the read overhead for our main application container so recreate this POD with an additional sidecar container that will run along with the application container and print to the STDOUT by running the command tail -f /var/log/elastic-app.log. You can use busybox image for this sidecar container.

* sidecar container running as expected.
* YAML file updated with the new container?

**Solution:**

From the question, we understand the ask is to get the logs from the container without creating a overhead. So let’s introduce a new container called sidecar which will pull the logs from the elastic app.log where the application sends the data to.

From the screen shot, we notice the application logs are sent to elastic-app.log file. Therefore, we will pull the latest logs from the log file by creating a new sidecar container.

Text

Description automatically generated

Text

Description automatically generated

**Question 11:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1.

A pod called color-app-cka13-arch has been created in the default namespace. This pod logs can be accessed using kubectl logs -f color-app-cka13-arch command from the student-node. It is currently displaying Color is pink output. Update the pod definition file to make use of the environment variable with the value - green and recreate this pod.

* pod updated?

**Solution:**

Lets switch to the cluster and display the logs from the container.

Text

Description automatically generated

Then, as per the requirement, we changed the environment variable to green.

Graphical user interface, application

Description automatically generated

Force delete the pod and recreate it. So that the logs display as green now.

Text

Description automatically generated

**Question 12:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3.

A pod called logger-cka03-arch has been created in the default namespace. Inspect this pod and save ALL INFO and ERROR's to the file /root/logger-cka03-arch-all on the student-node.

* Task completed?

**Solution:**

Question is straight forward, so lets use the below command to write the info and errors

Kubectl logs logger-cka03-arch | grep ‘INFO|ERROR’ > /root/logger-cka03-arch-all

**Question 13:**

For this question, please set the context to cluster1 by running:

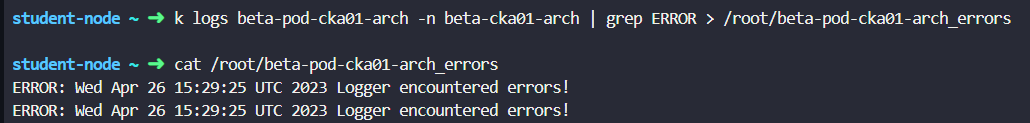
kubectl config use-context cluster1

A pod named beta-pod-cka01-arch has been created in the beta-cka01-arch namespace. Inspect the logs and save all logs starting with the string ERROR in file /root/beta-pod-cka01-arch\_errors on the student-node.

* task completed?

**Solution:**

Similar to the previous question, the question is to write the errors into a file. So use the below command



**Question 14:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

 Create a generic secret called db-user-pass-cka17-arch in the default namespace on cluster1 using the contents of the file /opt/db-user-pass on the student-node

* db-user-pass-cka17-arch secret created?

**Solution:**

Switch to the cluster and view the file to get the contents of the file.

Text

Description automatically generated

Create a secret using the imperative commands and supply the contents as a key value pair.

Text

Description automatically generated

To validate, describe the content and decode it using a base 64 format.

Text

Description automatically generated

**Question 15:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Decode the existing secret called beta-sec-cka14-arch created in the beta-ns-cka14-arch namespace and store the decoded content inside the file /opt/beta-sec-cka14-arch on the student-node.

* secret decoded?

**Solution:**

Using the documentation [link](https://kubernetes.io/docs/tasks/configmap-secret/managing-secret-using-kubectl/#decoding-secret), decode the secret name.

Text

Description automatically generated

# Troubleshooting

**Question 1:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

There is some issue on the student-node preventing it from accessing the cluster3 Kubernetes Cluster.

Troubleshoot and fix this issue. Make sure that you are able to run the kubectl commands (For example: kubectl get node --context=cluster3) from the student-node.

The kubeconfig for all the clusters is stored in the default kubeconfig file: /root/.kube/config on the student-node.

* + cluster3 accessible from student-node?

**Solution:**

Execute the kubectl command to get nodes or pods and see if it is accessible. As cluster 3 is inaccessible from student-node, lets check the configuration file directly to see if anything is suspicious.

* cat /root/.kube/config
* vi /root/.kube/config

Text

Description automatically generated

Its straightforward to see the control plane node is incorrect, where the default port listens to 6443. Lets correct it and access the nodes again.

student-node ~ ✦ ➜ k get nodes --context=cluster3

NAME STATUS ROLES AGE VERSION

cluster3-controlplane Ready control-plane,master 73m v1.24.1+k3s1

**Question 2:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

A new service account called thor-cka24-trb has been created in cluster1. Using this service account, we are trying to list and get the pods and secrets deployed in default namespace. However, this service account is not able to perform these operations.

Look into the issue and apply the appropriate fix(es) so that the service account thor-cka24-trb can perform these operations.

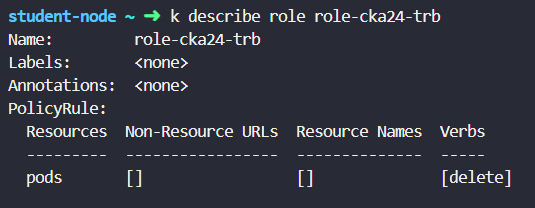
* Issue resolved?

**Solution:**

As the service accounts are associated to roles. Lets first check the roles bounded to the service account and role bindings associated with this.

Text

Description automatically generated



From the description of the role, we see the service account has got delete permission from pods.

Text

Description automatically generated

Lets edit the permission and update them to get and list the pods and secrets.

Validate the same by describing the role.

Text

Description automatically generated

Authenticate as service account and check if the SA has got right permission.

Text

Description automatically generated

**Question 3:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

A YAML template for a Kubernetes deployment is stored at /root/app-cka07-trb.yaml. However, creating a deployment using this file is failing. Investigate the cause of the errors and fix the issue.

Make sure that the pod is in running state once deployed.

Note: Do not to make any changes in the template file.

Verify the deployment.

**Solution:**

Trying to create the deployment with the given file says that the namespace is missing, it hasn’t been created. Therefore, lets create the namespace and deploy the file again.

A screenshot of a computer

Description automatically generated

Validate the deployment.

Text

Description automatically generated

**Question 4:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

demo-pod-cka29-trb pod is stuck in a Pending state, look into issue to fix the same, Make sure pod is in Running state and stable.

* Fixed the issues?
* ssh cluster1-controlplane pod is in running state?

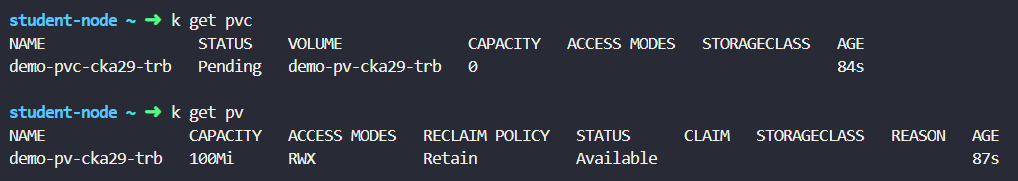
**Solution:**

Lets check the status of the pod and check the logs and events to understand more on the nature of the problem.

A picture containing text

Description automatically generated

Events gave us a clue that the problem is something with persistent volume or its claims. So checking the PVC gives the pending state which is not bound to existing PV.



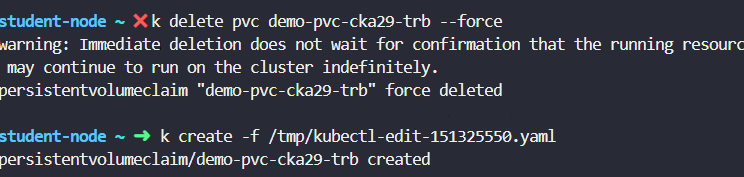
Now, let’s edit the pvc “demo-pvc-cka29-trb” and check few items between PV and PVC

* + Access modes.
  + Request limits
  + Capacity. etc

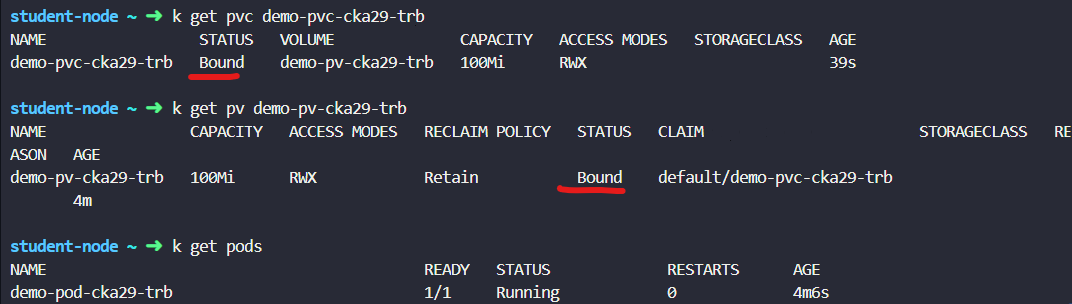
Text

Description automatically generated

By the look of access modes defined in the PVC is not inline with the PV (It should be ReadWriteMany). Therefore, the same has been corrected on the PVC.



Which has fixed the problem.



**Question 5:**

For this question, please set the context to cluster4 by running:

kubectl config use-context cluster4

The pink-depl-cka14-trb Deployment was scaled to 2 replicas however, the current replicas is still 1.

Troubleshoot and fix this issue. Make sure the CURRENT count is equal to the DESIRED count.

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

* CURRENT count is equal to the DESIRED count?

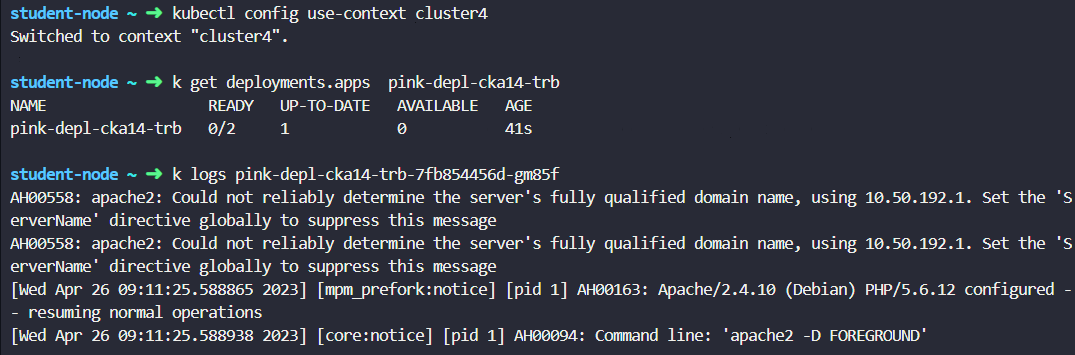
**Solution:**

Let us get the deployment status and check the logs and events by switching to cluster4 controlplane

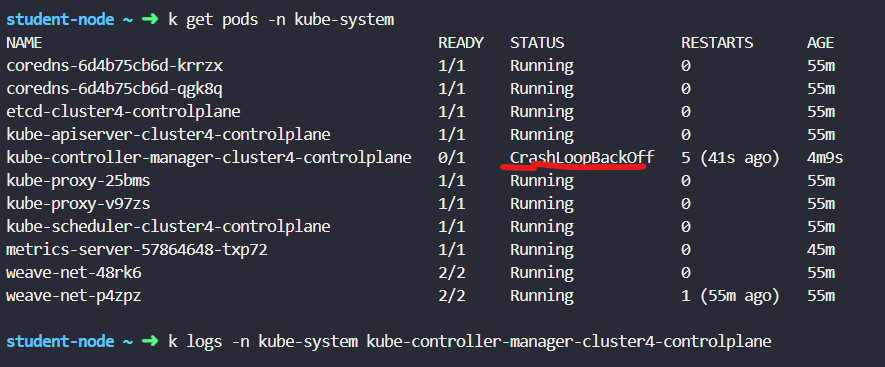
Text

Description automatically generated

As scheduling the pod to a specific node is the responsibility of the Kubernetes controller manager, lets check the status of the static pods on the kube-system namespace first to see if its up and running.



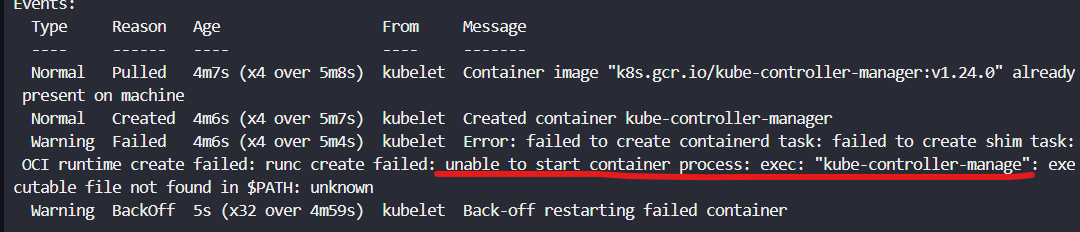
We could see the controller manager static pod is not up is in a state of crashloopback.



Events of the static pods could be checked using the command.

* k describe pods -n kube-system kube-controller-manager-cluster4-controlplane

From the events, we are able to see the container is unable to start.



Now, lets view the manifest file for the controller manager located in the path /etc/Kubernetes/manifests/\* in the controlplane node.

A screenshot of a computer

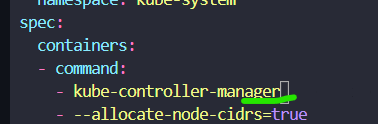
Description automatically generated with medium confidence

It is evident that the command has incorrect name of the controller manager.

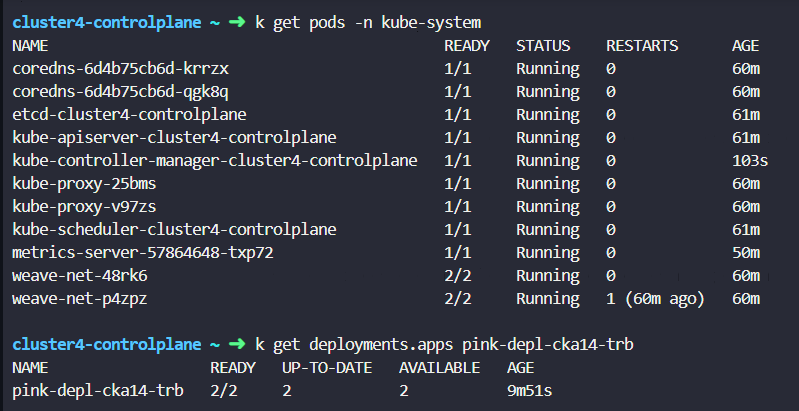
Text

Description automatically generated

This has been corrected now and saved which will fix the pod running in kube-system namespace.



Now, the pods are up and running which had scheduled the deployment (2/2)



**Question 6:**

For this question, please set the context to cluster2 by running:

kubectl config use-context cluster2

The yello-cka20-trb pod is stuck in a Pending state. Fix this issue and get it to a running state. Recreate the pod if necessary.

Do not remove any of the existing taints that are set on the cluster nodes.

* Node taints unchanged?
* pod is running?

**Solution:**

Looking at the events of the pod gives us an insight that the problem is on taints or tolerations.

A screenshot of a computer

Description automatically generated with medium confidence

Lets describe the node taints and view them.

Text

Description automatically generated

Now, with the taints we have on nodes lets validate the tolerations on the pod level.

Text

Description automatically generated

As taints and tolerations did not match, lets correct the value as per the taints on the tolerations.

Text

Description automatically generated

The pod has been recreated with the new toleration values and it is up and running.

Text

Description automatically generated

**Question 7:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

There is a deployment called nodeapp-dp-cka08-trb created in the default namespace on cluster1. This app is using an ingress resource named nodeapp-ing-cka08-trb.

From cluster1-controlplane host we should be able to access this app using the command: curl <http://kodekloud-ingress.app>. However, it is not working now. Troubleshoot and fix the issue.

Note: You should be able to ssh into the cluster1-controlplane using ssh cluster1-controlplane command.

* Fixed the issue?
* App is accessible?

**Solution:**

The solution approach for this problem should be.

* To check the deployment logs, events
* Ingress rules, ports used.
* Validate the ports used in the pods.
* Host name and path used, etc.

From the screen shot below, by describing the ingress, it clearly says the default backend endpoint does not exist and the backend is incorrect.

Text

Description automatically generated

Then the service responsible to call the pod has been listed out to get the backend name in the ingress resources.

Text

Description automatically generated

The host path and backend names along with the port has been corrected in the ingress resource yaml file and saved.

Text

Description automatically generated

Now when the host name is called it prints the hello world as expected.

Text

Description automatically generated

**Question 8:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

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.

Fixed the issue?

Pod Running?

**Solution:**

Deployment status says its pending.

Text

Description automatically generated

Events says that the OCI runtime creation has failed as the problem is somewhere with volume mounts

A picture containing text

Description automatically generated

Text

Description automatically generated

By looking at the pod definition, there are two different issues, we could see.

* 1. Problem is with the command where “-c” is missing.
  2. Nginx-config volumes.

Lets try to fix one by one.

Text

Description automatically generated

Text

Description automatically generated

* Update the command with ‘-c’ as it is missing.
* Reviewing the configuration map referred in the pod definition file. It is understood that the key value “nginx.conf” was not called out properly as a subPath in the Pod definition file. To understand more about the subPath, refer the [documentation.](https://kubernetes.io/docs/concepts/storage/volumes/#using-subpath)

This has been corrected now and the pod started to run.

Text

Description automatically generated

**Question 9:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

There is a deployment called nginx-dp-cka04-trb which has been used to deploy a static website. The access to this website can be tested by running: curl <http://kodekloud-exam.app:30002>. However, it is not working at the moment.

Troubleshoot and fix it.

* Website is working?

**Solution:**

Let us approach the problem by hitting the status of the static website first and see if it returns the expected results.

Screen shot shows that the connection was refused.

Text

Description automatically generated

Now, by looking at the events we could see the configuration maps defined in the deployment is not found.

Text

Description automatically generated

Therefore, correcting the same in the deployment definition file.

Text

Description automatically generated

Now we could see the pod is up and running.

Text

Description automatically generated

Application is up and running now.

Text

Description automatically generated

Text

Description automatically generated

**Question 10:**

For this question, please set the context to cluster2 by running:

kubectl config use-context cluster2

We recently deployed a DaemonSet called logs-cka26-trb under kube-system namespace in cluster2 for collecting logs from all the cluster nodes including the controlplane node. However, at this moment, the DaemonSet is not creating any pod on the controlplane node.

Troubleshoot the issue and fix it to make sure the pods are getting created on all nodes including the controlplane node.

* DaemonSet is able to run the PODs on all nodes?

**Solution:**

Now, as the problem states that the pods are not scheduled on the control plan node. Lets look at the node in which the daemonset pod is hosed on.

From the screen shot, it states the daemon set pods are in node01.

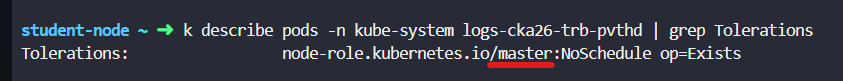
A screenshot of a computer

Description automatically generated with medium confidence

The reason could be due to the taints set on the control plane node. Lets review and apply the tolerations on the daemon set pods. Text

Description automatically generated

Tolerations set on the pods does not match with the taints on the node.

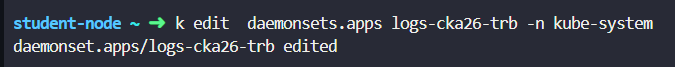


Now lets edit the daemon set with new toleration key value

Text

Description automatically generated

Now the daemon set pods are created successfully.



**Question 11:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

A template to create a Kubernetes pod is stored at /root/red-probe-cka12-trb.yaml on the student-node. However, using this template as-is is resulting in an error.

Fix the issue with this template and use it to create the pod. Once created, watch the pod for a minute or two to make sure its stable i.e, it's not crashing or restarting.

Make sure you do not update the args: section of the template.

* Template is fixed and applied?
* POD is stable?
* Template is not manipulated?

**Solution:**

To understand the problem, lets try creating the yaml file and observe the error.

A screenshot of a computer

Description automatically generated with medium confidence

From the error message, we understood the problem is with the livenessprobe. Now, lets check the official [documentation](https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/) to see if the syntax is appropriate in the yaml file.

Text

Description automatically generated

**Issue 1:** httpGet is inappropriate in the livenessProbe at this stage, as the probe is trying to execute the healthcheck of the application. To execute the command, the correct syntax is to use ‘exec’ in place of httpGet. So lets replace it.

**Issue 2:** When we read the command carefully, it instructs the container to sleep for 3 seconds, whereas the initial delay seconds which means the no. of seconds the livenessprobe has to wait when the container is up to initiate the first probe request is mentioned as 1 second. Due to which the container continuously crashes when the probe request is kick started after 1 second.

In the question, they mentioned not to modify the args section, therefore we could increase the initial delay seconds > 3. So here lets make the initial delay seconds to 5 and save the container.

Text

Description automatically generated

The pod is up and running now.

Text

Description automatically generated

**Question 12:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

The db-deployment-cka05-trb deployment is having 0 out of 1 PODs ready.

Figure out the issues and fix the same but make sure that you do not remove any DB related environment variables from the deployment/pod.

* DB deployment is fixed?

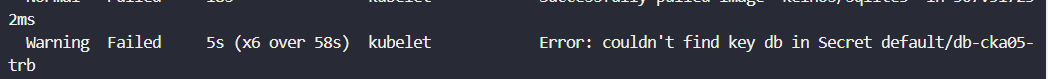
**Solution:**

From the screen shot we could see the pod is in container config error.

Text

Description automatically generated

By looking at the events, we could see the key is unavailable in the secret.



At this stage, we have to compare each and every secret keys and values mentioned in the secret name and compare the values in the pod definition.

Text

Description automatically generated

Text

Description automatically generated

All the secret keys and values has been fixed and the pod is up and running.

**Question 13:**

For this question, please set the context to cluster4 by running:

kubectl config use-context cluster4

There is a pod called pink-pod-cka16-trb created in the default namespace in cluster4. This app runs on port tcp/5000 and it is exposed to end-users using an ingress resource called pink-ing-cka16-trb in such a way that it is supposed to be accessible using the command: curl <http://kodekloud-pink.app> on cluster4-controlplane host.

However, this is not working. Troubleshoot and fix this issue, making any necessary to the objects.

Note: You should be able to ssh into the cluster4-controlplane using ssh cluster4-controlplane command.

* App is accessible?

**Solution:**

As the pod is exposed through a service. The primary rule is to validate if the endpoints are configured. The Pod IP matches with the service end points.

Issue 1: From the screen shot, we could see the ports are configured as UDP instead of TCP. So lets correct that.

Text

Description automatically generated

Then let’s try to hit the application from cluster 4. From the error message it says it could not resolve the host with the given name. So lets check if the DNS deployments are is up and running. If yes, lets view the events, logs of the DNS pods.

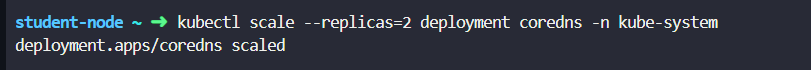
As we could see the coreDNS deployment is not scaled yet. Let’s scale them using an imperative command.

Text

Description automatically generated

Text

Description automatically generated



This brought up the core DNS pods up and running which resolved the issue.

**Question 14:**

For this question, please set the context to cluster1 by running

kubectl config use-context cluster1

It appears that the black-cka25-trb deployment in cluster1 isn't up to date. While listing the deployments, we are currently seeing 0 under the UP-TO-DATE section for this deployment. Troubleshoot, fix and make sure that this deployment is up to date.

* Issue fixed?
* Pods of the deployment is up-to-date?

**Solution:**

 By checking the deployment status, it says the up-to-date pods are 0, whereas the expected outcome is 1.

Text

Description automatically generated

Lets describe and check the status of the deployment. It says the deployment is paused, due to which the up-to-date field is 0.

Now lets resume the deployment using an imperative command.

A screenshot of a computer

Description automatically generated with medium confidence

Now the deployment up-to-date state is 1.

Text

Description automatically generated

**Question 15:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

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.

* POD(s) are running and stable?
* Issues are fixed?

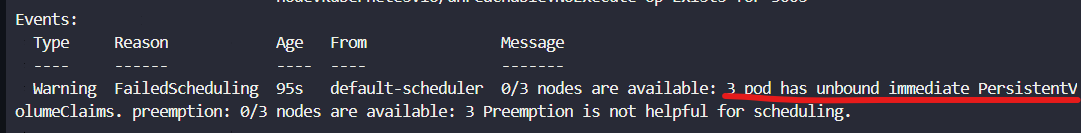
**Solution:**

As a first step, lets review the logs and events

Text

Description automatically generated

Issue 1: Events says the persistent volume is unbound with Persistent volume claims.



The same has been validated by executing imperative commands.

A screenshot of a computer

Description automatically generated

The persistent volume claim is requesting for 150 Mi whereas the persistent volume itself has a capacity of 100 Mi. So we have to reduce the resource request in PVC to 100 Mi to fix the issue.

Text

Description automatically generated

Graphical user interface, application

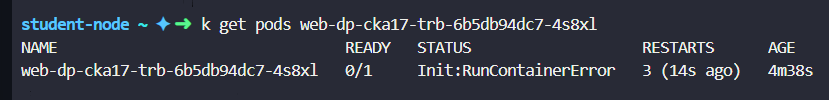
Description automatically generated

That has fixed the issue and bounded the PVC with PV.

Text

Description automatically generated

Issue 2: Looking at the pod status it says the container status is in error.



Events says the problem is with the type on a command /bin/bsh\.

Text

Description automatically generated with low confidence

The same has been corrected as /bin/sh

Text

Description automatically generated

Which had fixed the deployment issue.

Text

Description automatically generated

Let us validate that by getting the pod status.

Text

Description automatically generated

**Question 16:**

For this question, please set the context to cluster4 by running:

kubectl config use-context cluster4

On cluster4 we are having some weird issue where we are intermittently getting below error while running kubectl commands. The connection to the server cluster4-controlplane:6443 was refused - did you specify the right host or port?

Whenever you get this error, you can wait for 10-15 seconds to make kubectl command work again, but it will come again after few second

We also noticed that kube-controller-manager-cluster4-controlplane pod is restarting continuously. Look into the issue and troubleshoot the same.

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

* Issue is fixed?

**Solution:**

 As mentioned, the problem is intermittent. Therefore, lets try to get the pods using the kubectl command under the kube-system namespace. Although the status of the pod is running. The no. of restarts is 2 which means the pods kube-apiserver & kube-controller are not stable. The probes are making the container to restart.

Text

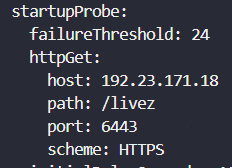
Description automatically generated

So lets view the static pod yaml file in the cluster 4 master node. From the pod definition, we see the port number is mentioned as 6444 which is incorrect. The api server will listen to the port 6443.

Text

Description automatically generated

This has been fixed now by updating the pod definition file. Now lets wait for few more minutes to have the pod up and running and wait for some more time to observe if the pod restarts again.



Pod is up and running, the issue has been solved.

Text

Description automatically generated

**Question 17:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

We deployed an app using a deployment called web-dp-cka06-trb. it's using the httpd:latest image. There is a corresponding service called web-service-cka06-trb that exposes this app on the node port 30005. However, the app is not accessible!

Troubleshoot and fix this issue. Make sure you are able to access the app using curl <http://kodekloud-exam.app:30005> command.

* Webapp is accessible?
* Issue 1

**Solution:**

Let us try to access the application first and observe the error message. It says the connection refused.

Text

Description automatically generated

As the pod is exposed through a service externally, our thumb rule is to validate if the endpoints are configured properly and the IP address in the endpoint matches with the pod’s own IP address. If not, check the labels used in the service and the pods.

Issue 1: From the screen shot, we see the labels does not match between the service configuration and the pod definition file. Lets fix that.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Issue 2: Now lets try accessing the application once. The error still persists. The pod status shows as pending.

A screenshot of a computer

Description automatically generated with medium confidence

By looking at the events, it shows the problem is with the PVC which is not found. While looking at the pod definition file, its understood the PVC name was configured incorrectly. Lets get that corrected in the pod definition file.

A screenshot of a computer screen

Description automatically generated

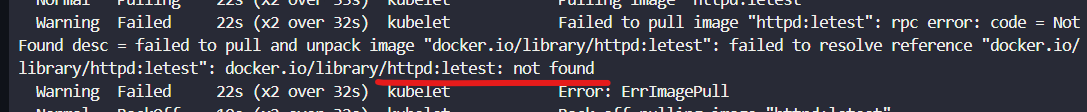
While viewing the pod status, it shows the pod status is still not up and running.

Text

Description automatically generated

Issue 3: By looking at the events, it states the container name is not found in the docker library.

From the look of the name, it looks like a typo. Lets correct it in the pod definition file.



All the issues have been fixed and the pod is up and running.

Text

Description automatically generated with medium confidence

**Question 18:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

One of the nginx based pod called cyan-pod-cka28-trb is running under cyan-ns-cka28-trb namespace and it is exposed within the cluster using cyan-svc-cka28-trb service.

This is a restricted pod so a network policy called cyan-np-cka28-trb has been created in the same namespace to apply some restrictions on this pod.

Two other pods called cyan-white-cka28-trb1 and cyan-black-cka28-trb are also running in the default namespace.

The nginx based app running on the cyan-pod-cka28-trb pod is exposed internally on the default nginx port (80).

Expectation: This app should only be accessible from the cyan-white-cka28-trb1 pod.

Problem: This app is not accessible from anywhere.

Troubleshoot this issue and fix the connectivity as per the requirement listed above.

Note: You can exec into cyan-white-cka28-trb and cyan-black-cka28-trb pods and test connectivity using the curl utility.

You may update the network policy, but make sure it is not deleted from the cyan-ns-cka28-trb namespace.

* App accessible from cyan-white-cka28-trb pod?
* App NOT accessible from cyan-black-cka28-trb pod?
* Network Policy still in use

**Solution:**

As a first step, lets try to understand and summarize our understanding from the question.

* One pod exist in cyan\* namespace exposed with a service internally.
* Network policy exist on the same namespace.
* Couple of other pods in a different namespace is trying to access this pod.
* Our goal is to make the white pod access the cyan pod in a different namespace and not to make the black pod accessible.

Now, lets jump to solve the problem. First, switch to the cluster and check if the endpoints are mapped properly between the service and the pods. Then, check the IP address in the endpoints with the pod’s IP address.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated with medium confidence

Lets describe the default network policies and observe the restrictions, port information, ingress configurations, etc.

Issue 1: While describing the network policy, we noticed the allowed ingress traffic is defined to the port 8080/TCP whereas the cyan white pod is in the port 80. Our goal is to allow the traffic from the port 80 to cyan-pod-cka28-trb. Let us edit it first.

A screenshot of a computer

Description automatically generated with medium confidence

Issue 2: If we look at the above screen shot closer, the ingress traffic is allowed only from the namespace selector “default”. Here our goal is to filter based on the pod labels to allow white pods for ingress. So let us update the network policies to allow the ingress traffic from the white pod in the default namespace.

Text

Description automatically generated

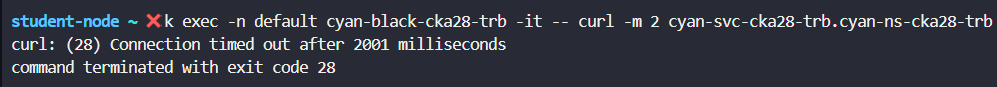
Let us now validate to see if white pod is able to access the pod *cyan-pod-cka28-trb*. This is done by executing a simple command from the white pod.

Text

Description automatically generated

***NOTE:*** If you take a closer look at the command to validate this, we called the pod name with namespace name, as the white pod exist in a different namespace.

Our second level of validation is to execute from black pod and ensure if the cyan-pod-cka28-trb is not accessible. As per the screen shot, its proven that from the black pod We are not able to access cyan-pod-cka28-trb



**Question 19:**

For this question, please set the context to cluster4 by running:

kubectl config use-context cluster4

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.

* Issues fixed?
* grey-cka21-trb POD is in running state?

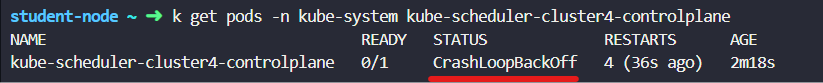
**Solution:**

Lets check the status of the pod first.

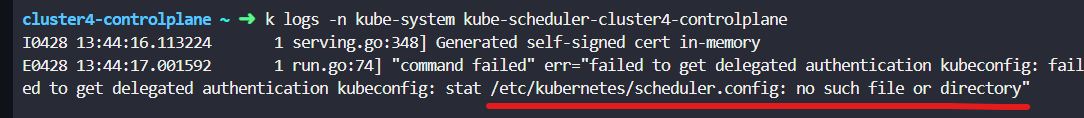
Text

Description automatically generated

As the responsibility of scheduling the pod is with kube scheduler, lets check the pod status in kube-system namespace. As we suspected, the pod status is crashloopbackoff.



Looking at the logs, we could see the scheduler.config does not exist.



The kubeconfig field had incorrect file name, which has been corrected.

Text

Description automatically generated

Now lets wait for the scheduler pod to be up and running.

Text

Description automatically generated

# Scheduling

**Question 1:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

In the dev-wl07 namespace, one of the developers has performed a rolling update and upgraded the application to a newer version. But somehow, application pods are not being created.

To get back the working state, rollback the application to the previous version .

After rolling the deployment back, on the controlplane node, save the image currently in use to the /root/rolling-back-record.txt file and increase the replica count to the 5.

You can SSH into the cluster1 using ssh cluster1-controlplane command.

* rolling back successful?
* image saved to the file?
* Replica set to 5?

**Solution:**

First, lets switch the context to the given cluster & the namespace to execute the imperative commands quickly. Now lets check the status of the pod to understand if the container creation is under trouble.

Text

Description automatically generated

By checking the history of the deployment rollout, we are able to notice there were two revisions to the deployment.

Lets try to rollout the deployment to the initial version to get back the pods to running status.

Text

Description automatically generated

As we could see the image name in the initial rollout version in the below screen shot

Text

Description automatically generated

Now the next ask is to write the image to a file and increase the replica by 5.

* Copy the image name and write it into the desired file.
* Use imperative commands to scale up the replica to 5

Text

Description automatically generated

**Question 2:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

One of our applications runs on the cluster3-controlplane node. Due to the possibility of traffic increase, we want to scale the application pods to loadbalance the traffic and provide a smoother user experience.

cluster3-controlplane node has enough resources to deploy more application pods. Scale the deployment called essports-wl02 to 5.

* Deployment scaled to 5?

**Solution:**

The solution is straightforward to use the imperative commands to increase the replica count by 5. Refer the screen shot and understand how it was done.

Text

Description automatically generated

**Question 3:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

A manifest file is available at the /root/app-wl03/ on the student-node node. There are some issues with the file; hence couldn't deploy a pod on the cluster3-controlplane node.

After fixing the issues, deploy the pod, and it should be in a running state.

NOTE: - Ensure that the existing limits are unchanged

* Pod app-wl03 is created with given limits?
* Pod is running?

**Solution:**

Creating the available file in the student-node would give us an idea about the failure. By the looks of the failure logs it looks like the resource requests were not under a defined limits (request : 1Gi, limits: 100 Mi).

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

**Question 4:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

The deployment called trace-wl08 inside the test-wl08 namespace on cluster1 has undergone several, routine, rolling updates and rollbacks.

Inspect the revision 2 of this deployment and store the image name that was used in this revision in the /opt/trace-wl08-revision-book.txt file on the student-node.

* Deployment is running?
* Revision 2 image saved to file?

**Solution:**

Lets look at the number of revisions the deployment had undergone by looking at the rollout history command.

Text

Description automatically generated

As the history did not provide the change cause. Let’s rollout the image to revision 2 to get the image name. then, describe the image name and write the same to a text file.

Text

Description automatically generated

**Question 5:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

We have deployed a simple web application called frontend-wl04 on cluster1. This version of the application has some issues from a security point of view and needs to be updated to version 2.

Update the image and wait for the application to fully deploy.

You can verify the running application using the curl command on the terminal:

student-node ~ ➜ curl <http://cluster1-node01:30080>  
<!doctype html><title>Hello from Flask</title><bodystyle="background: #2980b9;"></body><divstyle="color: #e4e4e4;  
 text-align: center;  
 height: 90px;  
 vertical-align: middle;"><h1>Hello from frontend-wl04-84fc69bd96-p7rbl!</h1><h2>Application Version: v1  
 </h2></div>

student-node ~ ➜  Version 2 Image details as follows:

1. Current version of the image is `v1`, we need to update with the image to kodekloud/webapp-color:v2.

2. Use the imperative command to update the image.

* Deployment is running with the new image?

**Solution:**

Simple two step approach to solve the problem is to

* Set the image to v2 using imperative command.
* Try accessing the application using the curl command to ensure the application load in version v2.

Text

Description automatically generated

**Question 6:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

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

* Deployment is running?

**Solution:**

Use imperative command to scale up the replica count to 2.

Text

Description automatically generated

Text

Description automatically generated

**Question 7:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

One of our Junior DevOps engineers have deployed a pod nginx-wl06 on the cluster3-controlplane node. However, while specifying the resource limits, instead of using Mebibyte as the unit, Gebibyte was used.

As a result, the node doesn't have sufficient resources to deploy this pod and it is stuck in a pending state. Fix the units and re-deploy the pod (Delete and recreate the pod if needed).

* Pod is running?
* Fix the memory?

**Solution:**

The pod definition needs a small correction in request🡪memory to have the units in Mi instead of Gi. Update it in the pod as shown below which made the pod available.

Text

Description automatically generated with medium confidence

Text

Description automatically generated

# Storage

**Question 1:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

There is a requirement to share a volume between two containers that are running within the same pod. Use the following instructions to create the pod and related objects:

  - Create a pod named grape-pod-cka06-str.

- The main container should use the nginx image and mount a volume called grape-vol-cka06-str at path /var/log/nginx.

- The sidecar container can use busybox image, you might need to add a sleep command to this container to keep it running. Next, mount the same volume called grape-vol-cka06-str at the path /usr/src.

- The volume should be of type emptyDir.

* containers of the pod use correct images?
* main container uses correct mount?
* sidecar container uses correct mount?
* volume created correctly?
* Pod is running?

**Solution:**

The ask is to create two containers in a same pod which is called a multi container pods in Kubernetes. Let’s kick start this by running an imperative command to create one container and output it to a temp file using –dry-run=client.

Text

Description automatically generated

Now lets edit the yaml file and add all the required configurations as per the question.

* Add another side car container to the pod using busybox image and make it to sleep for some time.
* Create a volume with emptyDir {} by following the syntax in the official documentation.
* Use the volume name to mount them in a different path given in the question.



Text

Description automatically generated

Text

Description automatically generated

Pod is up and running with given configuration.

Text

Description automatically generated

**Question 2:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

There is a persistent volume named apple-pv-cka04-str. Create a persistent volume claim named apple-pvc-cka04-str and request a 40Mi of storage from apple-pv-cka04-str PV.

The access mode should be ReadWriteOnce and storage class should be manual.

* Task Completed?

**Solution:**

Validate if the persistent volume is created first. Observe the capacity, access modes carefully.

Text

Description automatically generated

 Create the persistent volume claim by following the k8s documentation. Ensure the accessmode, capacity is inline with the PV.

Text

Description automatically generated

Now the status of the PV and PVC are bound.

A screenshot of a computer

Description automatically generated

**Question 3:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

Create a storage class with the name banana-sc-cka08-str as per the properties given below:

- Provisioner should be kubernetes.io/no-provisioner,

  - Volume binding mode should be WaitForFirstConsumer.

- Volume expansion should be enabled.

* banana-sc-cka08-str storage class created?

**Solution:**

The question is to create a storage class with a given properties. So lets directly use this resource [documentation](https://kubernetes.io/docs/concepts/storage/storage-classes/#the-storageclass-resource) to create the storage class

Text

Description automatically generated

Text

Description automatically generated

**Question 4:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

A persistent volume called papaya-pv-cka09-str is already created with a storage capacity of 150Mi. It's using the papaya-stc-cka09-str storage class with the path /opt/papaya-stc-cka09-str.

Also, a persistent volume claim named papaya-pvc-cka09-str has also been created on this cluster. This PVC has requested 50Mi of storage from papaya-pv-cka09-str volume.

Resize the PVC to 80Mi and make sure the PVC is in Bound state.

* papaya-pvc-cka09-str PVC resized?

**Solution:**

By describing the persistent volume we could see the status, access modes and capacity.

Text

Description automatically generated

Pvc exist with the similar configuration. Lets resize as per the question.

Text

Description automatically generated

PVC has been edited to 80 Mi and saved.

Text

Description automatically generated

# Service Networking

**Question 1:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

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

* Pod: nginx-resolver-cka06-svcn created
* Service DNS Resolution recorded correctly
* Pod DNS resolution recorded correctly ?
* "nginx-resolver-cka06-svcn" pod exposed using "nginx-resolver-service-cka06-svcn" ?

**Solution:**

First, let us create a pod and expose the pod internally using a service. (Exposing the pod internally means the service type should be Cluster IP). Use the imperative commands to create the same.

Text

Description automatically generated

Now when the service and pods are created, we have to validate the same by reviewing the endpoints, labels, IP addresses of the pod, etc.

Text

Description automatically generated

Next part of the question asks us to look up the service and store the results in a file.

NOTE: We have to be very careful while performing nslook up on the service and the pods as the FQDN differs for both service and pods.

* To perform a nslookup on a given service. Use the command.

kubectl exec -i -t dnsutils -- nslookup <service-name>.<namespace>.cluster.local

* To perform a nslookup on a given pod. Use the command.

kubectl exec -i -t dnsutils -- nslookup <pod\_name>.local

Refer the official [documentation](https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/#are-dns-endpoints-exposed) for more details. When we are able to display the output in the console, lets redirect it to a file as shown in the screen shot.

Text

Description automatically generated

**Question 2:**

For this question, please set the context to cluster3 by running:

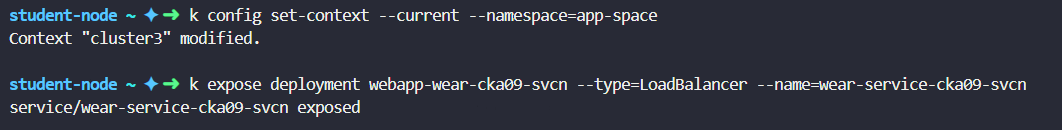
kubectl config use-context cluster3

Create a loadbalancer service with name wear-service-cka09-svcn to expose the deployment webapp-wear-cka09-svcn application in app-space namespace.

* "wear-service-cka09-svcn" created in "app-space" namespace?
* Type: LoadBalancer?
* Deployment 'webapp-wear' exposed?

**Solution:**

Straight forward approach to use imperative command and expose the deployment with a service type = LoadBalancer.



Once the service is exposed, lets validate the integration between the pod and service by validating the endpoints, IP address, Labels etc.

Text

Description automatically generated

**Question 3:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

John is setting up a two-tier application stack that is supposed to be accessible using the service curlme-cka01-svcn. To test that the service is accessible, he is using a pod called curlpod-cka01-svcn. However, now, he is unable to get any response from the application.

Troubleshoot and fix this issue so the application stack is accessible.

While you may delete and recreate the service curlme-cka01-svcn, please do not alter it in anyway.

* curlpod-cka01-svcn can access "curlme-cka01-svcn" pod?
* curlme-cka01-svcn properties not changed?

**Solution:**

*<Screen shots are not available for the given question>*

From the question, we can get a gist of the problem. The issue would be mostly on the integration between the pod and service. So, validate the below.

* + If the endpoint is configured properly?
  + Pod Labels match with the service pod selector field.
  + IP address in the end point matches with the pod’s IP address.

The problem was with the endpoint configuration, due to which we were unable to get any response from the application. To fix the problem, delete and recreate the service to use right pod endpoint.

**Question 4:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

**Part I:**

Create a ClusterIP service (i.e) service-3421-svcn which should expose the pods namely pod-23 and pod-21 with port set to 8080 and targetport to 80.

**Part II:**

Store the pod names and their ip addresses from all namespaces at /root/pod\_ips\_cka05\_svcn where the output is sorted by their IP's.

Please ensure the format as shown below:

 POD\_NAME IP\_ADDR  
pod-1ip-1pod-3ip-2pod-2ip-3...

**Solution:**

**Answers: Part 1**

The question is but tricky where the cluster IP service should be configured with two pod endpoints. We cannot achieve this by simple exposing the pod internally through a service.

This is possible by following the below steps.

* + Try to find out the commonalities between the pod 23 ad 21 in terms of labels.
  + Create a service with the identified pod selector fields.
  + Mention the right port and target port in the service.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Describe the service and check if the endpoints are configured for both the pods.

Text

Description automatically generated

**Answers: Part 2**

* + First get the pod displayed in JSON format.
  + Locate the IP fields in the JSON file.
  + Write a JSON Path query to get the desired pod IP’s.
  + Display the results in the terminal and review it.
  + Redirect the output to the given file name.

Output:

kubectl get pods -A -o=custom-columns='POD\_NAME:metadata.name,IP\_ADDR:status.podIP' --sort-by=.status.podIP > /root/pod\_ips\_cka05\_svcn

**Question 5:**

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

Create a pod with name tester-cka02-svcn in dev-cka02-svcn namespace with image registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3. Make sure to use command sleep 3600 with restart policy set to Always.

Once the tester-cka02-svcn pod is running, store the output of the command nslookup kubernetes.default from tester pod into the file /root/dns\_output on student-node.

* 'dev-cka02-svcn' namespace exists?
* 'tester-cka02-svcn' pod exists in dev-cka02-svcn namespace?
* correct image used?
* Restart policy set to "Always"?
* Command "sleep 3600" specified?
* Correct dns output stored in '/root/dns\_output"?

**Solution:**

Check if the namespace in the questionnaire exist. If not, create one.

Run a tester pod with the given image name and the command to sleep the container.

Text

Description automatically generated

Pod is up and running.

Text

Description automatically generated

Logon to the tester pod and execute the nslookup command and store the same in the output file.

Text

Description automatically generated

**Question 6:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

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 now 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.

**Solution:**

Let’s explore the service created which connects to the local webserver within the Kubernetes. We have identified the endpoints are not configured, due to which the integration between the pod and service does not work.

So lets try to grep the Ip address and create an endpoint on the service.

Text

Description automatically generated

Now while describing the service, the endpoint displays the port and IP address combination.

Text

Description automatically generated

Then accessing the external webserver had succeeded.

**Question 7:**

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

There is a deployment nginx-deployment-cka04-svcn in cluster3 which is exposed using service nginx-service-cka04-svcn.

Create an ingress resource nginx-ingress-cka04-svcn to load balance the incoming traffic with the following specifications:

* PathType: "Prefix"
* Path: "/"
* Backend serviceName: "nginx-service-cka04-svcn"
* Backend servicePort: "80"
* "ssl-redirect" is set to "false"?
* "nginx-ingress-cka04-svcn" ingress resource created?
* ingress working?

**Solution:**

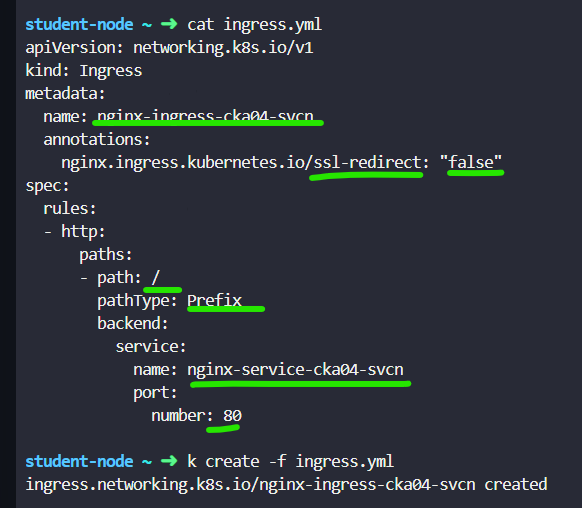
Execute a query to check if the deployment exists.

Question is to create an ingress resource for a deployment with a given configuration. At this stage, we could leverage the existing [documentation](https://kubernetes.io/docs/concepts/services-networking/ingress/#the-ingress-resource) to use the ingress resources syntax.

Modify the required fields as given and create the ingress resources for the deployment.

A screenshot of a computer

Description automatically generated



# Practical Exercise

|  |
| --- |
| SECTION: SCHEDULING |

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

Create a new deployment called ocean-tv-wl09 in the default namespace using the image kodekloud/webapp-color:v1.  
Use the following specs for the deployment:  
  
  
1. Replica count should be 3.  
  
2. Set the Max Unavailable to 40% and Max Surge to 55%.  
  
3. Create the deployment and ensure all the pods are ready.  
  
4. After successful deployment, upgrade the deployment image to kodekloud/webapp-color:v2 and inspect the deployment rollout status.  
  
5. Check the rolling history of the deployment and on the student-node, save the current revision count number to the /opt/revision-count.txt file.  
  
6. Finally, perform a rollback and revert back the deployment image to the older version.

* Deployment is running?
* Replica set to 3?
* maxSurge set to 55%?
* maxUnavailable set to 40%?
* Revision count stored in the file?
* Rolling back successful?

|  |
| --- |
| SECTION: SCHEDULING |

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

We have deployed a simple web application called frontend-wl04 on cluster1. This version of the application has some issues from a security point of view and needs to be updated to version 2.  
  
  
Update the image and wait for the application to fully deploy.  
  
  
You can verify the running application using the curl command on the terminal:

student-node ~ ➜ curl http://cluster1-node01:30080

<!doctype html>

<title>Hello from Flask</title>

<body style="background: #2980b9;"></body>

<div style="color: #e4e4e4;

text-align: center;

height: 90px;

vertical-align: middle;">

<h1>Hello from frontend-wl04-84fc69bd96-p7rbl!</h1>

<h2>

Application Version: v1

</h2>

</div>

student-node ~ ➜

Version 2 Image details as follows:

1. Current version of the image is `v1`, we need to update with the image to kodekloud/webapp-color:v2.  
2. Use the imperative command to update the image.

* Deployment is running with the new image?

|  |
| --- |
| SECTION: SCHEDULING |

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

A manifest file is available at the /root/app-wl03/ on the student-node node. There are some issues with the file; hence couldn't deploy a pod on the cluster3-controlplane node.  
  
After fixing the issues, deploy the pod, and it should be in a running state.  
  
  
NOTE: - Ensure that the existing limits are unchanged.

* Pod app-wl03 is created with given limits?
* Pod is running?

|  |
| --- |
| SECTION: SERVICE NETWORKING |

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Deploy a messaging-cka07-svcn pod using the redis:alpine image with the labels set to tier=msg.  
  
  
  
Now create a service messaging-service-cka07-svcn to expose the messaging-cka07-svcn application within the cluster on port 6379.

TIP: Use imperative commands.

* Pod Name: messaging-cka07-svcn
* Image: redis:alpine
* Labels: tier=msg
* Service: messaging-service-cka07-svcn
* Port: 6379
* Type: ClusterIP
* Use the right labels

|  |
| --- |
| SECTION: STORAGE |

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

A pod definition file is created at /root/peach-pod-cka05-str.yaml on the student-node. Update this manifest file to create a persistent volume claim called peach-pvc-cka05-str to claim a 100Mi of storage from peach-pv-cka05-str PV (this is already created). Use the access mode ReadWriteOnce.  
  
  
Further add peach-pvc-cka05-str PVC to peach-pod-cka05-str POD and mount the volume at /var/www/html location. Ensure that the pod is running and the PV is bound.

* PVC created ?
* peach-pvc-cka05-str PVC consumed by peach-pod-cka05-str pod?

|  |
| --- |
| SECTION: SERVICE NETWORKING |

For this question, please set the context to cluster3 by running:

kubectl config use-context cluster3

Create a deployment named hr-web-app-cka08-svcn using the image kodekloud/webapp-color with 2 replicas.  
  
  
  
Expose the hr-web-app-cka08-svcn as service hr-web-app-service-cka08-svcn application on port 30082 on the nodes of the cluster.  
  
The web application listens on port 8080.

* Name: hr-web-app-cka08-svcn
* Image: kodekloud/webapp-color
* Replicas: 2
* Name: hr-web-app-service-cka08-svcn
* Type: NodePort
* Endpoints: 2
* Port: 8080
* NodePort: 30082

Question 1:

For this question, please set the context to cluster1 by running:

kubectl config use-context cluster1

The green-deployment-cka15-trb deployment is having some issues since the corresponding POD is crashing and restarting multiple times continuously.

Investigate the issue and fix it, make sure the POD is in running state and its stable (i.e NO RESTARTS!).

* POD is stable now