Question 1: Creating Single Container Pods

Create a pod with the name of `kplabs-nginx`. The pod should be launched from an image of `mykplabs/kubernetes:nginx` . The name of the container should be `mycontainer`

Question 2: Multi-Container Pods

Create a Multi-Container POD with the name of `kplabs-multi-container`

There should be 3 containers as part of the pod. Name the first container as `first-container`, 2nd container as `second-container` and 3rd container as `third-container`

1st container should be launched from `nginx` image, second container should be launched from `mykplabs/kubernetes:nginx` image and third container from `busybox` image.

Connect to the first-container and run the following command: `apt-get update && apt-get install net-tools`

Connect to the third-container and identify the ports in which processes are listening. Perform `wget` command on those ports and check if you can download the HTML page.

Question 3: Commands and Arguments

Create a pod with the name of `kplabs-cmdargs`. The pod should be launched from an image of `busybox` . The name of the container should be `cmdcontainer`. Both the container image's CMD and ENTRYPOINT instruction should be overridden.

The container should start with `sleep` command and argument of `3600`

Question 4: Exposing Ports for PODS

Create a pod with the name of `kplabs-ports`. The pod should be launched from an image of `nginx`. The name of the container should be `nginx`. Expose Port `80` for the POD.

Question 5: CLI Documentation

1. List down all the available fields and it's associated description that we can include in a POD Manifest. Store data to `pod.txt`

2. List down all the fields & it's description that we can add under the metadata section under POD manifest. Store data to `pod-manifest.txt`

3. Matthew has realized that there is an option for `tolerationSeconds` under POD -> Spec -> Tolerations. Store the associated documentation for tolerationSeconds and store it under a file named tolerationSeconds.txt

Question 6: Arguments

Create a pod named `kplabs-logging`

The Pod should have a container running from the `nginx` image with the following arguments:

```
    - /bin/sh
    - -c
    - >
      i=0;
      while true;
      do
        echo "$i: $(date)" >> /var/log/1.log;
        echo "$(date) INFO $i" >> /var/log/2.log;
        i=$((i+1));
        sleep 1;
      done
```

Once POD is created, connect to the POD and verify the contents of `/var/log/1.log` and `/var/log/2.log`

Question 7: POD Troubleshooting

1. Create a POD with the name `kplabs-troubleshoot`. Launch it from `busybox` image.

2. Once launch, verify if you can see pod in "Ready" state.

3. If it's not in ready state, find out what can be the reason.

4. Edit the POD manifest to make sure busybox pod is available for at-least next 10 minutes.

Question 8: API Primitives

1. Create a proxy connection via `kubectl proxy --port 8080`

2. Verify from browser if you are able to see the list all the Kubernetes API's.

3. Find the list of resources under the `/api/v1`

4. Find the list of all the PODS running within your Kubernetes environment,

Question 9: Labels

Create a pod named `kplabs-label`. The pod should be launched from `nginx` image. The name of container should be `nginx-container`. Attach following label to the pod.

```
1. env=production
2. app=webserver
```

Question 10: Deployments

Creat a deployment named `kplabs-deployment`. The deployment should be launched from `nginx` image. The deployment should be three replicas. The selector should be based on the label of `app=nginx`

Question 11: Deployments - Rolling Updates and Rollbacks

1. Create a deployment named `kplabs-update`.

2. The deployment should be launched from `nginx` image.

3. There should be two replicas.

4. Verify the status of the deployment.

5. As part of rolling update, update the image to `nginx2:alpine`.

6. Verify the status of deployment.

7. Perform a rollback to the previous version and Verify the status of deployment.

Question 12: Readiness Probe

Launch a pod from the `nginx` image. Create a readiness probe for the pod.

Question 13: Labels and Selectors

1. Create a deployment named `kplabs-selector`.

2. The pods should be launched from `nginx` image.

3. The pods should only be launched in a node which has a label of `disk=ssd`

4. Observe the status of deployment.

5. Add the appropriate label to the worker node and then observe the status of the deployment.

Question 14: CronJob

Create a job named `kplabs-job`. The job should run every minute and should print out the current date.

Question 15: Service

1. Create a deployment named `kplabs-service`

2. The deployment should have three replicas and the image should be based on `nginx`

3. Create a service based on NodePort.

4. The service port should be 8080.

5. Verify if you are able to see the index.html of Nginx from service port.

Question 16: Service and Endpoints

1. Create a deployment named `deployment-manual`

Launch 3 replicas of `nginx` image. Create a service named service-manual.

Create an endpoint with the IP address of all the pods of `deployment-manual` and associate it with `service-manual`.

Verify the endpoints of the service to check if IP addresses have been populated.

</details>

Question 17: Load Balanacer Service

1. Create a deployment named `kplabs-dmz`

2. There should be two replicas of `nginx` image as part of the deployment.

3. Create a service name `service-elb` and it should be based on LoadBalancer type.

4. The Load Balancer should list on Port 80.

5. Once launched, verify that the nginx default page load when you open the Load Balancer IP in browser.

Question 18: Single Service Ingress

1. Create a pod named `single-pod`. Expose Port 80.

2, Create service named `single-service`. The service should listen on port of `8080`. It should direct traffic to single-pod

3. Create an ingress named `single-ingress`. Ingress should have following specification:

    i) Should direct all the traffic to single-service on the service port of 8080

4. Verify if the ingress is created.

5. Describe all the ingress rules.

6. Does the ingress have any IP address on "ADDRESS" field? If not, find out what can be the reason behind it?

Question 19: Named Based Virtual Hosting

1. Create an ingress which satisfies following requirement:

```
example.com   --|     Requests Redirects to      |  service1:8085
                |
test.com      --|     Requests Redirects to      |  service2:8089

Other Requests -|     Requests Redirects to      |  service3:9005
```

2. All the pods should be based on `nginx` image.

Question 20: Namespaces

1. Create a namespace named `kplabs-namespace`

2. Launch a POD from `nginx` image. Expose Port `8080`. Pod should be in kplabs-namespace

3. Verify the status of the pod with kubectl command.

Question 21: Resource Quota

Create a pod named `kplabs-quota`. The pod should have following configuration:

a. Should run with `nginx` image.

b. It should use maximum of 512 MiB of memory.

c. It should use maximum of 2 core CPU.

d. The POD should require a minimum of 128 MiB of memory before it is scheduled.

Question 22: Daemon Sets

1. Create a daemon set named `kplabs-daemonset`

2. The daemon set should launch 1 pod of `nginx` image in all the worker nodes.

3. Launch 1 more worker node in Digital Ocean and verify if daemon set is successful.

Question 23: Static Pods

1. Launch a K8s cluster based on minikube

2. Launch a static pod named `kplabs-static`. The pod should be launched from nginx image.

3. Verify the status of the pod.

Question 24: Node Selector

1. Create a Digital Ocean cluster which has two worker nodes.

2. The first worker node should have label of `worker=one`

3. The second worker node should have label of `worker=two`

4. Create a new pod named `workerone-pod`. The pod should only be launched in the first worker node.

5. Create a new pod named `workertwo-pod`. The pod should only be launched in the second worker node.

6. Write a kubectl command which shows list of available pods and it's associated worker nodes.

7. Verify if pods are created in the right worker nodes.

8. Remove all the labels and pods.

Question 25: Node Affinity - Required Scheduling

1. Create a Digital Ocean cluster which has two worker nodes.

2. Add the following label to worker node one `availabilityzone=az1`

3. Add the following label to worker node two `availabilityzone:=az2`

4. Create a pod named `kplabs-az1`. The pod should have an affinity were is only gets launched in node running in availability zone of az1

5. Create a pod named `kplabs-az2`. The pod can launch in any node which is not running in availability zone of az1

6. Create a pod named `kplabs-az3`. The pod should launch in any node which has is running in availabilityzone of az3

7. Verify the status of all the pods and check on which nodes they are running.

Question 26: Node Affinity - Preferred Scheduling

1. Create a pod based on Node Affinity.

2. Pod should be based on `nginx` image

3. Pod should be preferred to run on node which has following labels: `cpu=intel` or `cpu=amd`

4. If none of the nodes have the label, the pod should launch in any of the available node.

Question 27: Kubernetes Secrets

1. Create a new secret named `kplabs-secret`

2. The secret should have following data:

```
1.       username: admin
2.       password: password123
```

3. Mount the secret inside pod in form of environment variable in the following way:

a. username should be associated with environment variable of `DB_USERNAME` inside the pod.

a. password should be associated with environment variable of `DB_PASS` inside the pod.

4. Verify the same by connecting inside the container.

Question 28: Network Security Policies

Create three namespace named development, production, and security.

Launch a pod from `busybox` image in each of these namespace.

Pods within development namespace should not be able to communicate with pods in production namespace.

Pods within security should be able to communicate with all the pods.

Question 29: Network Security Policies

Create a namespace named `forensics`

All the pods within the forensics namespace should not be able to communicate outside world (egress isolation)

Create a pod named `investigator` in the default namespace.

Pods within forensics namespace should only allow connection from IP of the investigator pod.

Question 30: Linux Capabilities

1. Create a pod named kplabs-privileged.

2. All the containers inside the pod should be based on privileged mode.

3. Find out all the capabilities associated with the POD and store it in a file named capabilities.txt under /tmp directory.

Question 31: Creating User with Certificate Based Authentication

Alice is a new member of the DevOps team. She wants to have access to the K8s cluster to perform certain operations.

As part of security requirement, the authentication to K8s cluster should only be based on certificates. Perform all the necessary steps and generate the necessary kubeconfig file.

The output of following command should result in output:

```
kubectl --context=alice-context get pods
```

Question 32: Authorization

Alice has received a new task to deploy an application within the Kubernetes cluster. The application uses POD and Secret resources within the cluster. The application should be deployed only in the `production` namespace

Assign all the necessary permission to Alice to perform all operation on the resources required for application to be deployed within the production namespace.

Verify if everything works by creating a POD and Secret from the Alice user context in the production namespace.

Verify if everything works by creating a POD and Secret from the Alice user context in the default namespace.

Remove all the permission associated with the Alice user.

Question 33: Authorization

1. Create a cluster role in such way that it provides access to perform all read operations on resource PODS across all the namespaces.

2. Name of the role should be `kplabs-clusterrole`

3. Bind the cluster role to binding named `read-global-pods`

4. Associate the binding with the Alice user.

5. Create two namespace named `testing` and `production`.

6. Launch test pods in both the namespaces

6. With the context of Alice, verify if you are able to list the pods in the testing and production namespace.

Question 34: Secrets and Volumes

a. Create a secret name `app-creds` which has following data:

```
1.        appuser: dbreadonly
2.        apppass: myDBPassword#%
```

b. Create a pod with the name of `secret-pod`.

c. Mount the secret to the pod so that it is available in the path of `/etc/secret`

Question 35: ConfigMaps

1. Following is the configuration file for an application.

2. Create a ConfigMap based on the configuration file.

3. Mount the configmap in such a way that the configuration file is accessible in the location of `/etc/app.config`

```
        defaults
        mode http
        timeout connect 5000ms
        timeout client 50000ms
        timeout server 50000ms
```

Question 36: Pod Security Policies

Create a POD named kplabs-security with the following configuration:

a. The primary process should run with the userid of 1000

b. The primary group id should be 2000

Exam Preparation Practice Test

Question 37: POD Logging

Apply the following manifest to your K8s cluster:

`https://github.com/zealvora/myrepo/blob/master/demo-files/cka_logs.yaml`

Monitor the logs for all the containers which are part of the `counter2` pod. Extract log lines which have the number `02`. Write them to `/opt/kplabs-foobar`

Question 38: Daemonset

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 Daemon set to complete this task and use `kplabs-daemonset` as Daemonset's name.

Question 39: Init Container

Create a pod from the `busybox` image. Add an init container in such a way that it should create a file in the location of `/opt/myfile`. This file should be accessible from the nginx image as well. The name of pod should be base-pod

Question 40: Multi-Container Pod

Create a pod named kucc4 with a single container for each of the following images running inside (there may be between 1 and 4 images specified): nginx + redis + memcached + consul

Question 41: Node Selectors

Schedule a Pod as follows:

- Name: kplabs-selector
- Image:nginx
- Launch pod on the node which has a disktype of ssd.
- Make sure the Pod is in ready state.

Question 42: Deployment - Rolling Updates and Rollbacks

Create a deployment as follows

- Name : `nginx-app`
- Namespace: production
- Using container nginx with version `1.11.9-alpine`
- The deployment should contain 3 replicas
- Next, deploy the app 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 43: Service

Create and configure the service front-end-service so it's accessible through NodePort/ClusterIP and routes to the existing pod under the `nginx-app` deployment.

Question 44: PODS and Namespace

Create a Pod as follows:

- Name: jenkins
- Using image: jenkins
- In a new Kubernetes namespce named website-frontend

Question 45: Deployments

Create a deployment spec file that will ：

- Launch 7 replicas of the `redis` image with the label:app_env_stage=dev
- Deployment name: kua100201
- Save a copy of this spec file to /opt/KUAL00201/deploy_spec.yaml
- When you are done, clean up (delete) any new k8s API objects that you produced during this task

Question 46: Labels and Selectors

Create a file `/opt/KUCC00302/kucc00302.txt` that lists all pods in the front-end-service in the production namespace.

Question 47: Secrets

Create a Kubernetes Secret as follows:

Name: super-secret

Data of the secret:

```
credential=alice
username=bob
```

Create a Pod named `pod-secrets-via-file` using the `nginx` image which mounts a secret named `super-secret` at `/secrets`

Create a second Pod named `pod-secrets-via-env` using the `nginx` image, which exports credential and username as SUPERSECRET and USER environment variables.

Question 48: Volumes

Create a pod as following details:

- Name：non-persistent-redis
- Container image: redis
- Name-volume with name: cache-control
- Mount path: /data/redis
- It should launch in the pre-pod namespace and the volumes must not be persistent.

Question 49: Scaling the Deployment

Scale the `nginx-app` deployment to 6 pods.

Question 50: Metric Server

From the pods of the default namespace, identify the one which is taking the most CPU. Write the name of the pod to `/opt/cpu.txt`

Note: For your practice environment, go ahead and install metric server first.

Question 50: DNS

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

Use the utility nslookup to look up the DNS records of the service & pod and write the output to /opt/service.dns and /opt/pod.dns respectively

Question 51: Node Draining

Set the first node within your cluster to be unavailable and re-schedule all the pods running on it.

Question 52: Persistent Volumes

Create a persistent volume with name `app-config` of capacity 1Gi and access mode ReadWriteOnce. The type of volume is hostPath and its location is /opt/pvsort.txt

Question 53: Sorting Operation

Apply the following manifest file within your cluster:

```
https://github.com/zealvora/myrepo/blob/master/demo-files/pv-sorting.yaml
```

List all PVs sorted by the capacity and save the full kubectl output to /opt/my_volumes.txt

Use kubectl's own functionally for sorting the output and do not manipulate it any further

Question 54: Labels and Selectors

- Create a pod with the following specification:
- Name: kplabs-jenkins
- Image: jenkins
- The pod should have a label of env=development and org=kplabs

Question 55: Sorting Operation

List all the PVs sorted by the name and store the output to /opt/pv-sort-name.txt

Question 56: Deployments

Create a deployment with the following specification:

- Image: Nginx
- Name: kplabs-nginx-deploy
- Label: org=kplabs

- Replicas = 2

Question 57: Jobs

Create a job that will write "Hello CKA" in a total of 20 times with 2 parallelisms.

Question 58: POD Security Context

Create a pod named `pod-security`. The pod should be launched from the `busybox` image and it should run with the command of `ping 127.0.0.1` . The primary process should run with the UID of 1005. The pod should be launched in the namespace of `security`. Check the logs of the POD and output the log to the file /tmp/pod-security.txt

Question 59: Configuring Kubernetes Cluster

Launch two servers based on Ubuntu 18 image. Configure the Kubernetes cluster with kubeadm. Set 1 server as Master Node and 2nd server as a worker node. This question will be marked as complete once both the nodes's status is Ready.

Question 60: Taints

List all the nodes which do not contain any non-scheduling and not-reachable node. Write the output to /opt/nodes.txt

Question 61: Static Pods

Create a static pod on the worker node. The name should be kplabs-static and it should be launched from the nginx image.

Question 62: Taints

Add a taint to one of the worker node where key is mykey, value is mvalue and effect should be NoSchedule

Question 63: Toleration

Create a deployment named kplabs-tolerate. The deployment should be launched from `nginx` image and it should tolerate the taint of `mykey:myvalue:Noschedule`. Create 6 replicas of the deployment.

MUMSHAD EXAM:

Question 64: Deploy a pod

Deploy a pod named `nginx-pod` using the `nginx:alpine` image.

Once done, click on the `Next Question` button in the top right corner of this panel. You may navigate back and forth freely between all questions. Once done with all questions, click on `End Exam`. Your work will be validated at the end and score shown. Good Luck!

Name: nginx-pod

Image: nginx:alpine

Question 65: Deploy a `messaging` pod

Deploy a `messaging` pod using the `redis:alpine` image with the labels set to `tier=msg`.

Pod Name: messaging

Image: redis:alpine

Labels: tier=msg

Question 66: Namespace

Create a namespace named apx-x9984574.

Namespace: apx-x9984574

Question 67: Get the list of nodes in JSON

Get the list of nodes in JSON format and store it in a file at `/opt/outputs/nodes-z3444kd9.json`

Question 68: Service

Create a service `messaging-service` to expose the `messaging` application within the cluster on port `6379`.

Use imperative commands

Service: messaging-service

Port: 6379

Type: ClusterIp

Use the right labels

Question 69: Deployment

Create a deployment named `hr-web-app` using the image `kodekloud/webapp-color` with `2` replicas

Name: hr-web-app

Image: kodekloud/webapp-color

Replicas: 2

Question 70: Static Pod

Create a static pod named `static-busybox` on the master node that uses the `busybox` image and the command `sleep 1000`

Name: static-busybox

Image: busybox

Question 71: POD

Create a POD in the finance namespace named temp-bus with the image redis:alpine

`Name: temp-bus`

`Image Name: redis:alpine`

Question 72: Troubleshoot App

A new application `orange` is deployed. There is something wrong with it. Identify and fix the issue.

Question 73: Expose Service

Expose the `hr-web-app` as service `hr-web-app-service` application on port `30082` on the nodes on the cluster. The web application listens on port `8080`

Name: hr-web-app-service

Type: NodePort

Endpoints: 2

Port: 8080

NodePort: 30082

Question 74: JSON PATH

Use JSON PATH query to retrieve the `osImage`s of all the nodes and store it in a file `/opt/outputs/nodes_os_x43kj56.txt`

The `osImages` are under the `nodeInfo` section under `status` of each node.

Question 75: Persistent Volume

Create a Persistent Volume with the given specification.

`Volume Name: pv-analytics`

`Storage: 100Mi`

`Access modes: ReadWriteMany`

`Host Path: /pv/data-analytics`

Question 76: ETCD BACKUP

Take a backup of the etcd cluster and save it to /tmp/etcd-backup.db

Question 77: Volume

Create a Pod called `redis-storage` with image: `redis:alpine` with a Volume of type `emptyDir` that lasts for the life of the Pod. Specs on the right.

Pod named 'redis-storage' created

Pod 'redis-storage' uses Volume type of emptyDir

Pod 'redis-storage' uses volumeMount with mountPath = /data/redis

Question 78: Capability

Create a new pod called `super-user-pod` with image `busybox:1.28`. Allow the pod to be able to set system_time

The container should sleep for 4800 seconds

Pod: super-user-pod

Container Image: busybox:1.28

SYS_TIME capabilities for the conatiner?

Question 79: Persistent Volume and Persistent Volume Claim

A pod definition file is created at `/root/use-pv.yaml`. Make use of this manifest file and mount the persistent volume called `pv-1`. Ensure the pod is running and the PV is bound.

mountPath: /data persistentVolumeClaim Name: my-pvc

persistentVolume Claim configured correctly

pod using the correct mountPath

pod using the persistent volume claim?

Question 80: Deployment upgrade

Create a new deployment called nginx-deploy, with image nginx:1.16 and 1 replica. Record the version. Next upgrade the deployment to version 1.17 using rolling update. Make sure that the version upgrade is recorded in the resource annotation.

Deployment : nginx-deploy. Image: nginx:1.16

Image: nginx:1.16

Task: Upgrade the version of the deployment to 1:17

Task: Record the changes for the image upgrade

Question 81: New User Access

Create a new user called `john`. Grant him access to the cluster. John should have permission to `create, list, get, update and delete pods` in the `development` namespace . The private key exists in the location: `/root/john.key` and csr at `/root/john.csr`

CSR: john-developer Status:Approved

Role Name: developer, namespace: development, Resource: Pods

Access: User 'john' has appropriate permissions

Question 82: DNS Lookup

Create an nginx pod called `nginx-resolver` using image `nginx`, expose it internally with a service called `nginx-resolver-service`. 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/nginx.svc` and `/root/nginx.pod`

Pod: nginx-resolver created

Service DNS Resolution recorded correctly

Pod DNS resolution recorded correctly

Question 83: Static POD

Create a static pod on `node01` called `nginx-critical` with image `nginx`. Create this pod on node01 and make sure that it is recreated/restarted automatically in case of a failure.

Use `/etc/kubernetes/manifests` as the Static Pod path for example.

Kubelet Configured for Static Pods

Pod nginx-critical-node01 is Up and running

Question 84: Service Account

Create a new service account with the name `pvviewer`. Grant this Service account access to `list` all PersistentVolumes in the cluster by creating an appropriate cluster role called `pvviewer-role` and ClusterRoleBinding called `pvviewer-role-binding`.
Next, create a pod called `pvviewer` with the image: `redis` and serviceAccount: `pvviewer` in the default namespace

ServiceAccount: pvviewer

ClusterRole: pvviewer-role

ClusterRoleBinding: pvviewer-role-binding

Pod: pvviewer

Pod configured to use ServiceAccount pvviewer ?

Question 85: JSON PATH

List the `InternalIP` of all nodes of the cluster. Save the result to a file /root/node_ips

Answer should be in the format: `InternalIP of master`<space>`InternalIP of node1`<space>`InternalIP of node2`<space>`InternalIP of node3` (in a single line)

Question 86: Multi Container POD

Create a pod called `multi-pod` with two containers.
Container 1, name: `alpha`, image: `nginx`
Container 2: `beta`, image: `busybox`, command `sleep 4800`.

Environment Variables:
container 1:
`name: alpha`

Container 2:
`name: beta`

Pod Name: multi-pod

Container 1: alpha

Container 2: beta

Container beta commands set correctly?

Container 1 Environment Value Set

Container 2 Environment Value Set

Question 87: POD Security

Create a Pod called `non-root-pod` , image: `redis:alpine`
runAsUser: 1000

fsGroup: Pod `non-root-pod` fsGroup configured

Pod `non-root-pod` runAsUser configured2000

Question 88: Troubleshoot Network Policy

We have deployed a new pod called `np-test-1` and a service called `np-test-service`.
Incoming connections to this service are not working. Troubleshoot and fix it.
Create NetworkPolicy, by the name `ingress-to-nptest` that allows incoming connections to the service over port `80`

Important: Don't delete any current objects deployed.

Important: Don't Alter Existing Objects!

NetworkPolicy: Applied to All sources (Incoming traffic from all pods)?

NetWorkPolicy: Correct Port?

NetWorkPolicy: Applied to correct Pod?

Question 89: Taint Node

Taint the worker node node01 to be Unschedulable. Once done, create a pod called `dev-redis`, image `redis:alpine` to ensure workloads are not scheduled to this worker node. Finally, create a new pod called `prod-redis` and image `redis:alpine` with toleration to be scheduled on node01.

key:env_type, value:production, operator: Equal and effect:NoSchedule

Key = env_type

Value = production

Effect = NoSchedule

pod 'dev-redis' (no tolerations) is not scheduled on node01?

Create a pod 'prod-redis' to run on node01

Question 90: PODS and LABELS

Create a pod called `hr-pod` in `hr` namespace belonging to the `production` environment and `frontend` tier .
image: `redis:alpine`

Use appropriate labels and create all the required objects if it does not exist in the system already.

hr-pod labeled with environment production?

hr-pod labeled with frontend tier?

Question 91: Troubleshoot Kubeconfig File

A kubeconfig file called `super.kubeconfig` has been created in /root. There is something wrong with the configuration. Troubleshoot and fix it.

Question 91: Troubleshoot Deployment

We have created a new deployment called `nginx-deploy`. scale the deployment to 3 replicas. Has the replica's increased? Troubleshoot the issue and fix it.

deployment has 3 replicas

Question 92: POD

Deploy a pod named `nginx-448839` using the `nginx:alpine` image.

Once done, click on the `Next Question` button in the top right corner of this panel. You may navigate back and forth freely between all questions. Once done with all questions, click on `End Exam`. Your work will be validated at the end and score shown. Good Luck!

Name: nginx-448839

Image: nginx:alpine

Question 93: Deployment

Create a new Deployment named httpd-frontend with 3 replicas using image httpd:2.4-alpine.

Name: httpd-frontend

Replicas: 3

Image: httpd:2.4-alpine

Question 94: Namespace

Create a namespace named apx-z993845.

Question 95: POD

Deploy a messaging pod using the redis:alpine image with the labels set to tier=msg.

Pod Name: messaging

Image: redis:alpine

Labels: tier=msg

Question 96: Replicaset

A replicaset `rs-d33393` is created. However the pods are not coming up. Identify and fix the issue.

Once fixed, ensure the ReplicaSet has 4 `Ready` replicas

Question 97: Service

Create a service `messaging-service` to expose the `redis` deployment in the `marketing` namespace within the cluster on port `6379`.

Use imperative commands.

Service: messaging-service

Port: 6379

Use the right type of Service

Use the right labels

Question 98: Environment Variable

Update the environment variable on the pod webapp-color to use a green background.

Pod Name: webapp-color

Label Name: webapp-color

Env: APP_COLOR=green

Question 99: ConfigMap

Create a new ConfigMap named cm-3392845. Use the spec given below:

ConfigName Name: cm-3392845

Data: DB_NAME=SQL3322

Data: DB_HOST=sql322.mycompany.com

Data: DB_PORT=3306

Question 100: Secret

Create a new Secret named db-secret-xxdf with the data given(on the right).

Secret Name: db-secret-xxdf

Secret 1: DB_Host=sql01

Secret 2: DB_User=root

Secret 3: DB_Password=password123

Question 101: Capability

Update pod app-sec-kff3345 to run as Root user and with the SYS_TIME capability.

Pod Name: app-sec-kff3345

Image Name: ubuntu

SecurityContext: Capability SYS_TIME

Question 102: Logs

Export the logs of the `e-com-1123` pod to the file `/opt/outputs/e-com-1123.logs`

It is in a different namespace. Identify the namespace first.

Question 103: Persistent Volume

Create a Persistent Volume with the given specification.

Volume Name: pv-analytics

Storage: 100Mi

Access modes: ReadWriteMany

Host Path: /pv/data-analytics

Question 104: Network Policy

Create a redis deployment using the image redis:alpine with 1 replica and label app=redis. Expose it via a ClusterIP service called redis on port 6379. Create a new Ingress Type NetworkPolicy called redis-access which allows only the pods with label access=redis to access the deployment.

Image: redis:alpine

Deployment created correctly?

Service created correctly?

Network Policy allows the correct pods?

Network Policy applied on the correct pods?

Question 105: Multi Containers

Create a Pod called sega with two containers:

1. Container 1: Name tails with image busybox and command: sleep 3600.

2. Container 2: Name sonic with image nginx and Environment variable: NGINX_PORT with the value 8080.

Container Sonic has the correct ENV name

Container Sonic has the correct ENV value

Container tails created correctly?

Question 106: Deployment

Create a deployment called my-webapp with image: nginx, label tier:frontend and 2 replicas. Expose the deployment as a NodePort service with name front-end-service , port: 80 and NodePort: 30083.

Deployment my-webapp created?

image: nginx

Replicas = 2 ?

service front-end-service created?

service Type created correctly?

Correct node Port used?

Question 107: Taint

Add a taint to the node `node01` of the cluster. Use the specification below:

key:`app_type`, value:`alpha` and effect:`NoSchedule`
Create a pod called `alpha`, image:`redis` with toleration to node01

node01 with the correct taint?

Pod alpha has the correct toleration?

Question 108: Affinity

Apply a label app_type=beta to node node02. Create a new deployment called beta-apps with image:nginx and replicas:3. Set Node Affinity to the deployment to place the PODs on node02 only

NodeAffinity: requiredDuringSchedulingIgnoredDuringExecution

node02 has the correct labels?

Deployment beta-apps: NodeAffinity set to requiredDuringSchedulingIgnoredDuringExecution ?

Deployment beta-apps has correct Key for NodeAffinity?

Deployment beta-apps has correct Value for NodeAffinity?

Deployment beta-apps has pods running only on node02?

Deployment beta-apps has 3 pods running?

Question 109: Ingress Resource

Create a new Ingress Resource for the service: my-video-service to be made available at the URL: http://ckad-mock-exam-solution.com:30093/video.

Create an ingress resource with host: ckad-mock-exam-solution.com

path:/video

Once set up, curl test of the URL from the nodes should be successful / HTTP 200

Question 110: ReadinessProbe

We have deployed a new pod called pod-with-rprobe. This Pod has an initial delay before it is Ready. Update the newly created pod pod-with-rprobe with a readinessProbe using the given spec

httpGet path: /ready

httpGet port: 8080

readinessProbe with the correct httpGet path?

readinessProbe with the correct httpGet port?

Question 111: LivenessProbe

Create a new pod called nginx1401 in the default namespace with the image nginx. Add a livenessProbe to the container to restart it if the command ls /var/www/html/probe fails. This check should start after a delay of 10 seconds and run every 60 seconds.

You may delete and recreate the object. Ignore the warnings from the probe.

Question 112: Job

Create a job called whalesay with image docker/whalesay and command "cowsay I am going to ace CKAD!".

completions: 10

backoffLimit: 6

restartPolicy: Never

This simple job runs the popular cowsay game that was modifed by docker...

Job "whalsay" uses correct image?

Job "whalesay" configured with completions = 10?

Job "whalesay" with backoffLimit = 6

Job run's the command "cowsay I am going to ace CKAD!"?

Job "whalesay" completed successfully?

Question 113: Multi-container POD

Create a pod called multi-pod with two containers.

Container 1: name: jupiter, image: nginx

Container 2: europa, image: busybox

command: sleep 4800

Environment Variables: Container 1: type: planet

Container 2: type: moon

Pod Name: multi-pod

Container 1: jupiter

Container 2: europa

Container europa commands set correctly?

Container 1 Environment Value Set

Container 2 Environment Value Set

Question 114: Persistent Volume

Create a PersistentVolume called custom-volume with size: 50MiB reclaim policy:retain, Access Modes: ReadWriteMany and hostPath: /opt/data

PV custom-volume created?

custom-volume uses the correct access mode?

PV custom-volume has the correct storage capacity?

PV custom-volume has the correct host path?