CKA MOCK EXAM 5.0

time:2 hour

**submission :output in form of screenshoots or in a file ..(choice is yours)**

**QUESTION 1**

1. There are various Pods in all namespaces. Write a command into **/opt/course/6/find\_pods.sh** which lists all Pods sorted by their **AGE** .

2. Write a second command into **/opt/course/6/find\_pods\_uid.sh** which lists all Pods sorted by field **metadata.uid** .

Use kubectl sorting for both commands

**QUESTION 2**

1. Create a yaml file called **nginx-dp.yaml** for a deployment of four replicas of nginx, listening on the container's port 80.
2. They should have the labels **role=webserver** and **app=nginx**
3. The deployment should be named **nginx-dp**.
4. Expose the deployment with a load balancer and use a curl statement on the IP address of the load balancer to export the output to a file titled output.txt.

**QUESTION 3**

Scale the deployment you just made, **nginx-dp,** down to 2 replicas

**QUESTION 4**

1. Create a yaml file called **db-secret.yaml** for a secret called **db-user-pass**.
2. The secret should have two fields: a username and password.
3. The username should be **master\_admin** and the password should be **imamazing**.

**QUESTION 5**

1. Create a configmap named **mustard** with two values **mustard=yellow** and

# mayo=white.

1. Create a pod with the latest busybox image running a sleep for 1 hour, where you have two environment variables taken from the above configmap and show them in the container.

**QUESTION 6**

1. Create any pod of your choice
2. Perform a backup of the etcd data store.
3. Delete the pod you created
4. Perform a restore of the etcd data store.

**QUESTION 7**

1. Create a new service account with the name **podviewer**.
2. Grant this Service account access to list all Pods in the cluster by creating an appropriate cluster role called **podviewer-role** and ClusterRoleBinding called **podviewer-role-binding**.

**QUESTION 8**

Create a pod with an **Init Container**

Configure a Pod that runs two containers.

The first container should create the **/ data/speedracer.txt**.

The second container should only start once this file has been created. The second container run the **sleep 3600** command as its task.

**QUESTION 9**

Create a new user **"joaquin".**

Grant him access to the cluster.

User **"joaquin"** should have permission to create, list, get, update and delete pods.

The private key exists at location:

**/root/joaquin/.key** and csr at **/root/joaquin.csr**

**QUESTION 10**

1. Create a YAML file to define a pod named test-pod that uses the **busybox:latest** image.
2. Ensure the pod is scheduled to run only on the node named **node01** (if not using

Killercoda, run on any node you have).

**QUESTION 11**

Create a secret named **danish-secret** with the following key value pairs, then verify that the secret was created with the correct data

# username=admin\_dan

1. **password=Passw0rd1**

**QUESTION 12**

1. In a namespace called **prototype,** user **sam** should be allowed to create and delete Pods, Deployments and StatefulSets in Namespace **prototype**
2. User **sam** should have view permissions (like the permissions of the default ClusterRole named view) only in **etcd-controlplane** and **kube-scheduler- controlplane** namespaces
3. Verify everything using **kubectl auth can-i**

**QUESTION 13**

1. Upgrade the Cluster (Master Node)
2. Make sure to first drain both Node and make it available after upgrade.
3. Note: You may upgrade the cluster to any version you choose.

**QUESTION 14**

Create a new namespace named **mayonnaise**.

Create a new network policy named **my-policy** in the **mayonnaise** namespace Requirements:

1. Network Policy should allow pods within **mayonnaise** to connect to each other only on port 80.
2. No other ports should be allowed

2. No PODs from outside **mayonnaise** should be able to connect to any pods inside

# mayonnaise

**QUESTION 15**

1. Create a nginx pod called **dns-resolver-pod** using image **nginx:latest**
2. Expose it internally with a service called **dns-res—svc**
3. Check if pod and service name are resolvable from within the cluster
4. Use the image: **busybox:1.28** for DNS lookup
5. Save the result in **/root/nginx.svc**

**QUESTION 16**

Create a static pod named **alpine-static** on the node **node01** that uses the image

# nginx:alpine

**QUESTION 17**

Create a Persistent Volume (PV) and a corresponding Persistent Volume Claim (PVC) and then use it in a Pod.

1.Create a Persistent Volume named my-pv with the following specifications:

Capacity: **1Gi**

Access Modes**: ReadWriteOnce**

HostPath: **/mnt/data**

Storage Class: **manual**

2.Create a Persistent Volume Claim named my-pvc with the following specifications:

Requested Storage: **1Gi**

Access Modes: **ReadWriteOnce**

Storage Class: **manual**

3.Create mount the volume at **/usr/share** Pod named **nginx-pod** that uses the **PVC my-pvc** to are**/nginx/html**.

4.Verify that the Pod is running and using the Persistent Volume.