Mock Exam

Q1) Deploy a **pod** named **cka1** in namespace **cncf** based on the image **nginx:1.16.0**

Q2) a) Create a 5-replica deployment with the name **cache** based on the **memcached** image

b)Expose the deployment as an internal only service using port & target-port of 11211

c) Print the endpoints object associated with that service in json format to a file **q2.json**

d) list all pods in that deployment only, sort the output by pod name, print the output to **q2.txt**

Q3) a) Create a **deployment** called cka3 that uses **busybox** as an **init container** to write the hostname of that **pod** to a file in a **non-persistent** volume. Then mount this file in a **nginx:latest** container At /usr/share/nginx/html/index.html

b) **Expose** the deployment using a NodePort service so that it can be accessed via a curl from your machine. Validate your deployment by using curl to access your service.

c) **Scale** up the deployment to **5** tracking all changes. Use curl to confirm your service has scaled up

d) Use a **busybox:1.28** pod to do a nslookup of the service using its DNS entry and write the outputs to a file

Q4) Create a yaml file called **db-secret.yaml** for a secret called **db-user-pass**.

The secret should have two fields: a **username** and **password**.

The username should be "**superadmin**" and the password should be "**imamazing**".

Q5) Create a deployment running **nginx**, mount a volume called "**hostvolume**" with a container volume mount at **/tmp** and mounted to the host at **/data (on a mac change this to a directory in your home directory)**. If the directory isn't there make sure it is created in the pod spec at run time.

Go into the container and create an empty file called "**my-doc.txt**" inside the **/tmp** directory. On the worker node

that it was scheduled to, go into the **/data** directory and output a list of the contents to list-output.txt showing

the file exists.

Q6) Create a secret that has the following username password data:

**username=missawesome**

**password=123kube321**

Create a pod running **nginx** that has access to those data items in a volume mount path at **/tmp/secret-volume**

log into the **nginx** pod you created and list the items and cat the output of the data items to a file "**credentials.txt**"

Q7) Create a yaml file called **nginx-deploy.yaml** for a deployment of three replicas of **nginx**, listening on the container's port **80**.

They should have the labels **role=webserver** and **app=nginx**. The deployment should be named **nginx-deploy**.

Expose the deployment with a load balancer and use a curl statement on the IP address of the load balancer.

**Note**: If running on your GCP cluster you won’t get an external IP provisioned so instead use a worker hostname/IP address and the NodePort port number generated.

to export the output to a file titled output.txt.

Q8) Perform an etcd backup **ON YOUR GCP CLUSTER** saving the snapshot to ~/mockexamsnapshotdb. Verify the snapshot has saved correctly.

**Answers:**

Q1) Deploy a **pod** named **cka1** in namespace **cncf** based on the image **nginx:1.16.0**

**A1.yaml**

apiVersion: v1

kind: Namespace

metadata:

creationTimestamp: null

name: cncf

spec: {}

status: {}

---

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

run: cka1

name: cka1

namespace: cncf

spec:

containers:

- image: nginx:1.16.0

name: cka1

resources: {}

dnsPolicy: ClusterFirst

restartPolicy: Never

status: {}

Q2) a) Create a 5-replica deployment with the name **cache** based on the **memcached** image

b)Expose the deployment as an internal only service using port & target-port of 11211

c) Print the endpoints object associated with that service in json format to a file **q2.json**

d) list all pods in that deployment only, sort the output by pod name, print the output to **q2.txt** (see <https://stackoverflow.com/questions/52957227/kubectl-command-to-list-pods-of-a-deployment-in-kubernetes>)

**A2-deploy.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

run: cache

name: cache

spec:

replicas: 5

selector:

matchLabels:

run: cache

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

run: cache

spec:

containers:

- image: memcached

name: cache

resources: {}

status: {}

Q3) a) Create a **deployment** called cka3 that uses **busybox** as an **init container** to write the hostname of that **pod** to a file in a **non-persistent** volume. Then mount this file in a **nginx:latest** container At /usr/share/nginx/html/index.html

b) **Expose** the deployment using a NodePort service so that it can be accessed via a curl from your machine. Validate your deployment by using curl to access your service.

c) **Scale** up the deployment to **5** tracking all changes. Use curl to confirm your service has scaled up

d) Use a **busybox:1.28** pod to do a nslookup of the service using its DNS entry and write the outputs to a file

**A3-deploy.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

run: cka3

name: cka3

spec:

replicas: 1

selector:

matchLabels:

run: cka3

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

run: cka3

spec:

containers:

- image: nginx:latest

name: cka3

volumeMounts:

- name: volume

mountPath: /usr/share/nginx/html/

initContainers:

- image: busybox

name: init

command: ["sh", "-c", "hostname > /mnt/index.html"]

volumeMounts:

- name: volume

mountPath: /mnt/

resources: {}

volumes:

- name: volume

emptyDir: {}

status: {}

**A3-svc.yaml**

apiVersion: v1

kind: Service

metadata:

creationTimestamp: null

labels:

run: cka3

name: cka3

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: cka3

type: NodePort

status:

loadBalancer: {}

**A3 (lookup pod example output)**

$ kubectl exec -ti busybox -- sh

/ # nslookup 192-168-43-25.default.pod.cluster.local

Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: 192-168-43-25.default.pod.cluster.local

Address 1: 192.168.43.25 192-168-43-25.cka3.default.svc.cluster.local

**A3 (lookup service example output)**

$ kubectl exec -ti busybox -- sh

/ # nslookup cka3

Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: cka3

Address 1: 10.106.191.142 cka3.default.svc.cluster.local

/ # nslookup cka3.default.svc.cluster.local

Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: cka3.default.svc.cluster.local

Address 1: 10.106.191.142 cka3.default.svc.cluster.local

Q4) Create a yaml file called **db-secret.yaml** for a secret called **db-user-pass**.

The secret should have two fields: a **username** and **password**.

The username should be "**superadmin**" and the password should be "**imamazing**".

**A4-db-secret.yaml**

apiVersion: v1

data:

password: aW1hbWF6aW5n

username: c3VwZXJhZG1pbg==

kind: Secret

metadata:

creationTimestamp: null

name: db-user-pass

Q5) Create a deployment running **nginx**, mount a volume called "**hostvolume**" with a container volume mount at **/tmp** and mounted to the host at **/data (on a mac change this to a directory in your home directory)**. If the directory isn't there make sure it is created in the pod spec at run time.

Go into the container and create an empty file called "**my-doc.txt**" inside the **/tmp** directory. On the worker node

that it was scheduled to, go into the **/data** directory and output a list of the contents to list-output.txt showing

the file exists.

**A5-deploy.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

run: nginx

name: nginx

spec:

replicas: 1

selector:

matchLabels:

run: nginx

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

run: nginx

spec:

containers:

- image: nginx

name: nginx

volumeMounts:

- name: hostvolume

mountPath: /tmp

volumes:

- name: hostvolume

hostPath:

path: /data

type: DirectoryOrCreate

Q6) a) Create a secret that has the following username password data:

**username=missawesome**

**password=123kube321**

b) Create a pod running **nginx** that has access to those data items in a volume mount path at **/tmp/secret-volume**

log into the **nginx** pod you created and list the items and cat the output of the data items to a file "**credentials.txt**"

**A6-create-secret**

$ kubectl create secret generic mymisssecret --from-literal=username=missawesome --from-literal=password=123kube321

**A6-pod.yaml**

apiVersion: v1

kind: Pod

metadata:

creationTimestamp: null

labels:

run: q6-pod

name: q6-pod

spec:

containers:

- image: nginx

name: q6-pod

resources: {}

volumeMounts:

- mountPath: /tmp/secret-volume

name: secret-vol

volumes:

- name: secret-vol

secret:

secretName: mymisssecret

dnsPolicy: ClusterFirst

restartPolicy: Never

status: {}

Q7) a) Create a yaml file called **nginx-deploy.yaml** for a deployment of three replicas of **nginx**, listening on the container’s port **80**.

They should have the labels **role=webserver** and **app=nginx**. The deployment should be named **nginx-deploy**.

b) 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.

**Note**: If running on your GCP cluster you won’t get an external IP provisioned so instead use a worker hostname/IP address and the NodePort port number generated.

**A7-deploy.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

creationTimestamp: null

labels:

app: nginx

role: webserver

name: nginx-deploy

spec:

replicas: 3

selector:

matchLabels:

app: nginx

role: webserver

strategy: {}

template:

metadata:

creationTimestamp: null

labels:

app: nginx

role: webserver

spec:

containers:

- image: nginx

name: nginx-deploy

ports:

- containerPort: 80

resources: {}

status: {}

**A7-svc.yaml**

apiVersion: v1

kind: Service

metadata:

creationTimestamp: null

labels:

app: nginx

role: webserver

name: nginx-deploy

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

app: nginx

role: webserver

type: LoadBalancer

status:

loadBalancer: {}

Q8) Perform an etcd backup **ON YOUR GCP CLUSTER** saving the snapshot to ~/mockexamsnapshotdb. Verify the snapshot has saved correctly.

**A8-backup-etcd**

$ sudo ETCDCTL\_API=3 etcdctl --endpoints 10.140.0.10:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot save ~/mockexamsnapshotdb

**A8-veryify-etcd**

$ sudo ETCDCTL\_API=3 etcdctl --endpoints 10.140.0.10:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key –write-out=table snapshot status ~/mockexamsnapshotdb