Services & Networking

Weight : 2

1) For this question, please set this context (In exam, diff cluster name)

```
kubectl config use-context kubernetes-admin@kubernetes
```

You have an existing Nginx pod named `nginx-pod` . Perform the following steps:

- Expose the `nginx-pod` internally within the cluster using a Service named `nginx-service` .
- Use `port forwarding` to service to access the Welcome content of nginx-pod using the `curl` command.

Solution:-

Step 1: expose nginx-pod svc `kubectl expose pod nginx-pod --name=nginx-service --port=80 --target-port=80 --type=ClusterIP`

Step 2: To verify run `kubectl port-forward service/nginx-service 8080:80`

Open another terminal and run `curl http://localhost:8080`

Weight : 5

2) For this question, please set this context (In exam, diff cluster name)

```
kubectl config use-context kubernetes-admin@kubernetes
```

Part I:

- Create a Kubernetes ClusterIP service named `nginx-service` . This service should expose to `nginx-deployment` , using port `8080` and target port `80`

Part II:

- Retrieve and store the IP addresses of the pods. Sort the output by their IP addresses in Ascending order and save it to the file `pod_ips.txt` in the following format:

```
IP_ADDRESS

127.0.0.1

127.0.0.2

127.0.0.3
```

Solution:-

Step 1: expose nginx-deployment

```
kubectl expose deployment nginx-deployment --name=nginx-service --port=8080 --target-port=80 --type=ClusterIP
```

Step 2: get pod IPs(in Ascending order) and store(with title IP_ADDRESS) it in a file pod_ips.txt

```
kubectl get pods -o wide
```

Weight : 6

3) For this question, please set this context (In exam, diff cluster name)

```
kubectl config use-context kubernetes-admin@kubernetes
```

Create a ReplicaSet named `dns-rs-cka` with `2` replicas in the `dns-ns` namespace using the image `registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3` and set the command to `sleep 3600` with the container named `dns-container` .

Once the pods are up and running, run the `nslookup kubernetes.default` command from any one of the pod and save the output into a file named `dns-output.txt` .

Solution:-

Step 1: create namespace

```
kubectl create ns dns-ns
```

Step 2: get deployment template using imperative way(Fast way)

```
kubectl create deploy dns-rs-cka --namespace=dns-ns --image=registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3 --replicas=2 --dry-run=client -o yaml > replicaset.yaml
```

Step 3: Update deployment template to ReplicaSet

a) From- `kind: Deployment` , To- `kind: ReplicaSet`

b) Replace container From- `name: jessie-dnsutils` , To- `name: dns-container`

c) Add `command: ["sleep", "3600"]` under container and delete this line `strategy: {}` , Final YAML :-

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  labels:
    app: dns-rs-cka
  name: dns-rs-cka
  namespace: dns-ns
spec:
```

```
  replicas: 2
  selector:
    matchLabels:
      app: dns-rs-cka
  template:
    metadata:
      labels:
        app: dns-rs-cka
    spec:
      containers:
      - image: registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3
        name: dns-container
        command: ["sleep", "3600"]
```

d) `kubectl apply -f replicaset.yaml`

Step 4: Get one pod name `kubectl get pod -n dns-ns`

Step 5: Now let's run nslookup command

`kubectl exec -n dns-ns "dns-rs-cka-8fh4f" -- nslookup kubernetes.default` Here, `"dns-rs-cka-8fh4f"` Pod

Threw:- `;; connection timed out; no servers could be reached`

`command terminated with exit code 1`

Step 6: looks like problem with the name resolution, let's check coredns pods and services are healthy

`kubectl get pods -n kube-system | grep coredns` 'coredns' Pods are Healthy

`kubectl get service -n kube-system` 'kube-dns' Service also looks good

a) Let's check any endpoints for kube-dns service

`kubectl get endpoints kube-dns -n kube-system`

O/p: `NAME       ENDPOINTS   AGE`

`kube-dns   <none>      15d`

Here we can observe thers is no endpoints for kube-dns service

Step 6: let's check again any problem with 'coredns' deployment and 'kube-dns' Service

`kubectl describe deployment coredns -n kube-system` Here, labels:- `k8s-app=kube-dns`

`kubectl describe service kube-dns -n kube-system` Here, `k8s-app=core-dns`

Step 7: let's update kube-dns service selector from- `k8s-app=core-dns` to- `k8s-app=kube-dns`

`kubectl patch service kube-dns -n kube-system -p '{"spec":{"selector":{"k8s-app": "kube-dns"}}}'`

a) Let's check again any endpoints for kube-dns service

```
kubectl get endpoints kube-dns -n kube-system
```

O/p:
```
NAME       ENDPOINTS   AGE
kube-dns   192.168.0.7:53,192.168.1.2:53,192.168.0.7:53 + 3 more...   15d
```

Now, looks good

Step 8: let's try to run nslookup command

```
kubectl exec -n dns-ns "dns-rs-cka-8fh4f" -- nslookup kubernetes.default
```

O/p:
```
Server:         10.96.0.10
Address:        10.96.0.10#53

Name:   kubernetes.default.svc.cluster.local
Address: 10.96.0.1
```

Now o/p looks good

Step 9: Save the output in a file dns-output.txt

```
kubectl exec -n dns-ns "dns-rs-cka-8fh4f" -- nslookup kubernetes.default > dns-output.txt
```

Weight : 4

4) For this question, please set this context (In exam, diff cluster name)

`kubectl config use-context kubernetes-admin@kubernetes`

Create a Deployment named `dns-deploy-cka` with `2` replicas in the `dns-ns` namespace using the image `registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3` and set the command to `sleep 3600` with the container named `dns-container` .

Once the pods are up and running, run the `nslookup kubernetes.default` command from any one of the pod and save the output into a file named `dns-output.txt` .

Solution:-

Step 1: create namespace

```
kubectl create ns dns-ns
```

Step 2: get deployment template using imperative way(Fast way)

kubectl create deploy dns-deploy-cka --namespace=dns-ns --image=registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3 --replicas=2 --dry-run=client -o yaml > deploy.yaml

Step 3: Update deployment template

b) Replace container From- name: jessie-dnsutils , To- name: dns-container

c) Add command: ["sleep", "3600"] under container , Final YAML :-

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dns-rs-cka
  name: dns-rs-cka
  namespace: dns-ns
spec:
  replicas: 2
  selector:
    matchLabels:
      app: dns-rs-cka
  template:
    metadata:
      labels:
        app: dns-rs-cka
    spec:
      containers:
      - image: registry.k8s.io/e2e-test-images/jessie-dnsutils:1.3
        name: dns-container
        command: ["sleep", "3600"]
```

d) kubectl apply -f deploy.yaml

Step 4: Get one pod name kubectl get pod -n dns-ns

Step 5: Now let's run nslookup command

kubectl exec -n dns-ns "dns-rs-cka-8fh4f" -- nslookup kubernetes.default Here, "dns-rs-cka-8fh4f" Pod

O/p: Server: 10.96.0.10

Address: 10.96.0.10#53

Name: kubernetes.default.svc.cluster.local

Address: 10.96.0.1

Now o/p looks good

Step 6: Save the output in a file dns-output.txt

kubectl exec -n dns-ns "dns-rs-cka-8fh4f" -- nslookup kubernetes.default > dns-output.txt

Weight : 5

5) For this question, please set this context (In exam, diff cluster name)

`kubectl config use-context kubernetes-admin@kubernetes`

There exists a deployment named `nginx-deployment` exposed through a service called `nginx-service` . Create an ingress resource named `nginx-ingress-resource` to efficiently distribute incoming traffic with the following settings: `pathType: Prefix` , `path: /shop` , Backend Service `Name: nginx-service` , Backend Service `Port: 80` , ssl-redirect should be configured as `false` .

Solution:-

create ingress resource using imperative way(fast way to create)

```
kubectl create ingress nginx-ingress-resource  --rule="/shop*=nginx-service:80"
--annotation=nginx.ingress.kubernetes.io/ssl-redirect="false"
```

Weight : 6

6) For this question, please set this context (In exam, diff cluster name)

`kubectl config use-context kubernetes-admin@kubernetes`

`my-app-deployment` deployed, and they are exposed through a service named `my-app-service` . Create a NetworkPolicy named `my-app-network-policy` to restrict incoming and outgoing traffic to these pods with the following specifications:

- Allow incoming traffic only from pods within the same namespace.
- Allow incoming traffic from a specific pod with the label "app=trusted."
- Allow outgoing traffic to pods within the same namespace.
- Deny all other incoming and outgoing traffic.

Solution:- Create network policy yaml file

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-app-network-policy
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
```

```
    - podSelector: {}
  - from:
    - podSelector:
        matchLabels:
          app: trusted
 egress:
  - to:
    - podSelector: {}
```

Weight : 4

7) For this question, please set this context (In exam, diff cluster name)

`kubectl config use-context kubernetes-admin@kubernetes`

Create a NodePort service named `app-service-cka` (with below specification) to expose the `nginx-app-cka` deployment in the `nginx-app-space` namespace.

- port & target port `80`
- protocol `TCP`
- node port `31000`

Solution:-

Step 1: Get service template

```
kubectl expose deployment nginx-app-cka --name=app-service-cka --type=NodePort --port=80 --target-port=80 --protocol=TCP -n nginx-app-space --dry-run=client -o yaml > svc.yaml
```

Step 2: add nodePort under ports section `nodePort: 31000`

Step 3: run kubeclt command `kubectl apply -f svc.yaml`

Weight : 4

8) For this question, please set this context (In exam, diff cluster name)

`kubectl config use-context kubernetes-admin@kubernetes`

Create a deployment named `my-web-app-deployment` using the Docker image `wordpress` with `2` replicas. Then, expose the `my-web-app-deployment` as a service named `my-web-app-service` , making it accessible on port `30770` on the nodes of the cluster.

Solution:-

Step 1: create ‘my-web-app-deployment’ deployment

```
kubectl create deployment my-web-app-deployment --image=wordpress --replicas=2
```

Step 2: create service template

```
kubectl expose deployment my-web-app-deployment --name=my-web-app-service --type=NodePort --port=80 --target-port=80 --dry-run=client -o yaml > svc.yaml
```

Step 3: add nodePort under ports section `nodePort: 30770`

Step 4: run kubeclt command `kubectl apply -f svc.yaml`

Weight : 4

9) For this question, please set this context (In exam, diff cluster name)

```
kubectl config use-context kubernetes-admin@kubernetes
```

Create an nginx pod named `nginx-pod-cka` using the `nginx` image, and expose it internally with a service named `nginx-service-cka` . Verify your ability to perform DNS lookups for the service name from within the cluster using the `busybox:1.28` image. Record the results in `nginx-service.txt` .

Solution:-

Step 1: create 'my-web-app-deployment' deployment

```
kubectl run nginx-pod-cka --image=nginx --restart=Never
```

Step 2: create 'nginx-service-cka' service

```
kubectl expose pod nginx-pod-cka --name=nginx-service-cka --port=80
```

Step 3: perform DNA lookup

```
kubectl run nslookup --image=busybox:1.28 --restart=Never --rm -it -- nslookup nginx-service-cka > nginx-service.txt
```
