Kubernetes Ecosystem Homework

Author: Bokotei Oleksandr

GitHub: https://github.com/nonamecoder2002/GLBaseCamp2021

[Base]:

Task1: “Canary deployment”

At first, let's prepare 2 versions of the app:

nc_serv

gcr.io / second-terra-315309 / nc_serv

Filter by name or tag

Name	Tags
6ad7d0242327	1.0.0
760b2be0cbf4	2.0.0

Then deploy nc_serv:1.0.0 using the following code:

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ncserv-depl
  labels:
    app: ncserv
    version: "1.0.0"
spec:
  replicas: 4
  selector:
    matchLabels:
      app: ncserv
  template:
    metadata:
      labels:
        app: ncserv
        version: "1.0.0"
    spec:
      containers:
      - name: ncserv
        image: gcr.io/second-terra-315309/nc_serv:1.0.0
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: ncserv-service
spec:
  type: LoadBalancer
  selector:
    app: ncserv
    version: "1.0.0"
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

The code above creates 4 pods that run nc_serv:1.0.0 & creates the ncserv-service that exposes the application via the external LoadBalancer. To verify that the deployment was successful, let's get the LoadBalancer external IP & make http request to that IP address:

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get svc -o wide
NAME             TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)        AGE     SELECTOR
kubernetes       ClusterIP      10.72.0.1     <none>          443/TCP        124m    <none>
ncserv-service   LoadBalancer   10.72.9.72    34.134.94.169   80:31386/TCP   2m24s   app=ncserv,version=1.0.0
```

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get ep
NAME             ENDPOINTS                                                     AGE
kubernetes       107.178.220.110:443                                           124m
ncserv-service   10.68.0.12:80,10.68.0.17:80,10.68.0.20:80 + 1 more...         2m32s
```

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get po -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP
ncserv-depl-f5968dd-k7shv     1/1     Running   0          63m   10.68.0.20
ncserv-depl-f5968dd-p64jc     1/1     Running   0          90m   10.68.0.12
ncserv-depl-f5968dd-pgjgf     1/1     Running   0          63m   10.68.0.17
ncserv-depl-f5968dd-wpvv4     1/1     Running   0          63m   10.68.0.22
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$
```

```
alex@DESKTOP-LBU2UOH:~$ while true; do curl -s  34.134.94.169:80; sleep 0.5; done
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
```

As we can see, the deployment is successful

Now let`s deploy 1 replica of nc_serv:2.0.0 using the code below & verify if the deployment is successful:

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ncserv-canary
  labels:
    app: ncserv
    version: "2.0.0"
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ncserv
  template:
    metadata:
      labels:
        app: ncserv
        version: "2.0.0"
    spec:
      containers:
      - name: ncserv
        image: gcr.io/second-terra-315309/nc_serv:2.0.0
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: ncserv-service
spec:
  type: LoadBalancer
  selector:
    app: ncserv
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

The code above creates 1 pod using nc_serv:2.0.0 image & deletes the “version” selector from the ncserv-service configuration

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get svc -o wide
NAME             TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)        AGE    SELECTOR
kubernetes       ClusterIP      10.72.0.1     <none>          443/TCP        146m   <none>
ncserv-service   LoadBalancer   10.72.9.72    34.134.94.169   80:31386/TCP   23m    app=ncserv
```

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get ep
NAME             ENDPOINTS                                                     AGE
kubernetes       107.178.220.110:443                                           146m
ncserv-service   10.68.0.12:80,10.68.0.17:80,10.68.0.20:80 + 2 more...         23m
```

```
alex@DESKTOP-LBU2UOH:~/kube_ecosystem$ kubectl get po -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP
OMINATED NODE   READINESS GATES
ncserv-canary-6c686d949d-b2wwt    1/1     Running   0          35s   10.68.0.23
none>           <none>
ncserv-depl-f5968dd-k7shv         1/1     Running   0          71m   10.68.0.20
none>           <none>
ncserv-depl-f5968dd-p64jc         1/1     Running   0          97m   10.68.0.12
none>           <none>
ncserv-depl-f5968dd-pgjgf         1/1     Running   0          71m   10.68.0.17
none>           <none>
ncserv-depl-f5968dd-wpvv4         1/1     Running   0          71m   10.68.0.22
none>           <none>
```

```
alex@DESKTOP-LBU2UOH:~$ while true; do curl -s  34.134.94.169:80; sleep 0.5; done
 Version 1.0.0
 Version 1.0.0
 Version 2.0.0
 Version 2.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 2.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
 Version 1.0.0
```

As we can see, approximately 20% of the traffic goes through the ncserv-canary & the ncserv-canary is working OK

Task 2: "Troubleshoot app-secret-env.yaml"

To make app-secret-env.yaml code work OK we need to create a secret with the appropriate keys:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret1
type: Opaque
data:
  username: dXNlcgo=
  password: dXNlcjEyMzQ=
```

Then send this secret to the kuber cluster:

```
alex@DESKTOP-LBU2UOH:~$ kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-vzhrq   kubernetes.io/service-account-token   3      22h
mysecret1             Opaque                                2      21m
alex@DESKTOP-LBU2UOH:~$
```

Now we can run app-secret-env.yaml

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl get po
NAME              READY   STATUS    RESTARTS   AGE
app-secret-env    1/1     Running   0          31s
```

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl logs app-secret-env
1:C 07 Jun 2021 11:44:22.823 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 07 Jun 2021 11:44:22.823 # Redis version=6.2.4, bits=64, commit=00000000,
1:C 07 Jun 2021 11:44:22.823 # Warning: no config file specified, using the d
1:M 07 Jun 2021 11:44:22.824 * monotonic clock: POSIX clock_gettime
1:M 07 Jun 2021 11:44:22.825 * Running mode=standalone, port=6379.
1:M 07 Jun 2021 11:44:22.825 # Server initialized
1:M 07 Jun 2021 11:44:22.826 * Ready to accept connections
```

To verify that`s everything is OK lets get inside the pod & echo the env vars:

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl exec -it app-secret-env -- sh
# echo $SECRET_USERNAME
user
# echo $SECRET_PASSWORD
user1234
#
```

Task 3: "Deploy & Expose app-multicontainer.yaml"

Using LoadBalancer:

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: multiconapp-depl
  labels:
    app: multiconapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: multiconapp
```

```yaml
  template:
    metadata:
      labels:
        app: multiconapp
    spec:
      volumes:
      - name: html
        emptyDir: {}
      containers:
      - name: 1st
        image: nginx
        volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
        ports:
        - containerPort: 80
      - name: 2nd
        image: debian
        volumeMounts:
        - name: html
          mountPath: /html
        command: ["/bin/sh", "-c"]
        args:
          - while true; do
              date >> /html/index.html;
              sleep 1;
            done
---
apiVersion: v1
kind: Service
metadata:
  name: multiconapp-service
spec:
  type: LoadBalancer
  selector:
    app: multiconapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl get svc
NAME                 TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes           ClusterIP      10.72.0.1     <none>         443/TCP        6h
multiconapp-service  LoadBalancer   10.72.8.73    35.226.59.10   80:32357/TCP   56m
```

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl get ep
NAME                 ENDPOINTS              AGE
kubernetes           107.178.220.110:443    6h
multiconapp-service  10.68.0.25:80          57m
```

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl get po -o wide
NAME                                 READY   STATUS    RESTARTS   AGE   IP
multiconapp-depl-5864b4fbdd-8t2ll    2/2     Running   0          83s   10.68.0.25
```

Not secure | 35.226.59.10

Sun Jun 6 19:36:09 UTC 2021 Sun Jun 6 19:36:10 UTC 2021 Sun Jun 6 19:36:12 UTC 2021 Sun Jun
19:36:19 UTC 2021 Sun Jun 6 19:36:20 UTC 2021 Sun Jun 6 19:36:21 UTC 2021 Sun Jun 6 19:36:22
2021 Sun Jun 6 19:36:29 UTC 2021 Sun Jun 6 19:36:30 UTC 2021 Sun Jun 6 19:36:31 UTC 2021 Su
19:36:38 UTC 2021 Sun Jun 6 19:36:39 UTC 2021 Sun Jun 6 19:36:40 UTC 2021 Sun Jun 6 19:36:41
2021 Sun Jun 6 19:36:48 UTC 2021 Sun Jun 6 19:36:49 UTC 2021 Sun Jun 6 19:36:50 UTC 2021 Su
19:36:57 UTC 2021 Sun Jun 6 19:36:58 UTC 2021 Sun Jun 6 19:36:59 UTC 2021 Sun Jun 6 19:37:00
2021 Sun Jun 6 19:37:07 UTC 2021 Sun Jun 6 19:37:08 UTC 2021 Sun Jun 6 19:37:09 UTC 2021 Su
19:37:16 UTC 2021 Sun Jun 6 19:37:17 UTC 2021 Sun Jun 6 19:37:18 UTC 2021 Sun Jun 6 19:37:19
2021 Sun Jun 6 19:37:26 UTC 2021 Sun Jun 6 19:37:27 UTC 2021 Sun Jun 6 19:37:28 UTC 2021 Su
19:37:35 UTC 2021 Sun Jun 6 19:37:36 UTC 2021 Sun Jun 6 19:37:37 UTC 2021 Sun Jun 6 19:37:38
2021 Sun Jun 6 19:37:45 UTC 2021 Sun Jun 6 19:37:46 UTC 2021 Sun Jun 6 19:37:47 UTC 2021 Su
19:37:54 UTC 2021 Sun Jun 6 19:37:55 UTC 2021 Sun Jun 6 19:37:56 UTC 2021 Sun Jun 6 19:37:57

Using NodePort:

The deployment template is the same. Only the service code is different:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: multiconapp-service
spec:
  type: NodePort
  selector:
```

```
    app: multiconapp
  ports:
    - protocol: TCP
      port: 80
      nodePort: 30001
```

But to make this work we need to create the firewall rule that allows TCP traffic to port 30001 on the node external IP:

noodle-cluster-nodeport-ext	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:30000-31000	Allow	1000	default

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl get no -o wide
NAME                                          STATUS   ROLES    AGE    VERSION           INTERNAL-IP   EXTERNAL-IP
RSION   CONTAINER-RUNTIME
gke-noodle-cluster-noodle-pool-bab3162f-l2jt   Ready    <none>   7h1m   v1.19.9-gke.1900   10.128.0.5    35.192.94.199
        containerd://1.4.3
```

Let`s connect to EXTERNAL-IP:30001:

Not secure | 35.192.94.199:30001

Apps Cloud Computing S...

Sun Jun 6 19:43:30 UTC 2021 Sun Jun 6 19:43:31 UTC 2021 Sun Jun 6 19:43:32 UTC 2021
19:43:39 UTC 2021 Sun Jun 6 19:43:40 UTC 2021 Sun Jun 6 19:43:41 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:43:49 UTC 2021 Sun Jun 6 19:43:50 UTC 2021 Sun Jun 6 19:43:51
Sun Jun 6 19:43:58 UTC 2021 Sun Jun 6 19:43:59 UTC 2021 Sun Jun 6 19:44:00 UTC 2021
19:44:07 UTC 2021 Sun Jun 6 19:44:08 UTC 2021 Sun Jun 6 19:44:09 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:44:17 UTC 2021 Sun Jun 6 19:44:18 UTC 2021 Sun Jun 6 19:44:19
Sun Jun 6 19:44:26 UTC 2021 Sun Jun 6 19:44:27 UTC 2021 Sun Jun 6 19:44:28 UTC 2021
19:44:35 UTC 2021 Sun Jun 6 19:44:36 UTC 2021 Sun Jun 6 19:44:37 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:44:45 UTC 2021 Sun Jun 6 19:44:46 UTC 2021 Sun Jun 6 19:44:47
Sun Jun 6 19:44:54 UTC 2021 Sun Jun 6 19:44:55 UTC 2021 Sun Jun 6 19:44:56 UTC 2021
19:45:03 UTC 2021 Sun Jun 6 19:45:04 UTC 2021 Sun Jun 6 19:45:05 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:45:13 UTC 2021 Sun Jun 6 19:45:14 UTC 2021 Sun Jun 6 19:45:15
Sun Jun 6 19:45:22 UTC 2021 Sun Jun 6 19:45:23 UTC 2021 Sun Jun 6 19:45:24 UTC 2021
19:45:31 UTC 2021 Sun Jun 6 19:45:32 UTC 2021 Sun Jun 6 19:45:33 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:45:41 UTC 2021 Sun Jun 6 19:45:42 UTC 2021 Sun Jun 6 19:45:43
Sun Jun 6 19:45:50 UTC 2021 Sun Jun 6 19:45:51 UTC 2021 Sun Jun 6 19:45:52 UTC 2021
19:45:59 UTC 2021 Sun Jun 6 19:46:00 UTC 2021 Sun Jun 6 19:46:01 UTC 2021 Sun Jun 6
UTC 2021 Sun Jun 6 19:46:09 UTC 2021 Sun Jun 6 19:46:10 UTC 2021 Sun Jun 6 19:46:11

[EXT]:

Let's use the same Deployment code as in the Task 3 but without any Service:

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: multiconapp-depl
  labels:
    app: multiconapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: multiconapp
  template:
    metadata:
      labels:
        app: multiconapp
    spec:
      volumes:
      - name: html
        emptyDir: {}
      containers:
      - name: 1st
        image: nginx
        volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
        ports:
        - containerPort: 80
      - name: 2nd
        image: debian
        volumeMounts:
        - name: html
```

```
          mountPath: /html
        command: ["/bin/sh", "-c"]
        args:
          - while true; do
              date >> /html/index.html;
              sleep 1;
            done
```

```
alex@DESKTOP-LBU2UOH:~$ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
multiconapp-depl-5864b4fbdd-6d5jr   2/2     Running   0          19m
alex@DESKTOP-LBU2UOH:~$
```

Now let's use port-forwarding to connect to the pod from localhost:

```
alex@DESKTOP-LBU2UOH:~/kuber_ecosystem$ kubectl port-forward multiconapp-depl-5864b4fbdd-6d5jr 8008:80
Forwarding from 127.0.0.1:8008 -> 80
Forwarding from [::1]:8008 -> 80
Handling connection for 8008
```

127.0.0.1:8008

Apps Cloud Computing S...

Mon Jun 7 08:38:29 UTC 2021 Mon Jun 7 08:38:30 UTC 2021 Mon Jun 7 08:38:31 UTC 2021 Jun 7 08:38:38 UTC 2021 Mon Jun 7 08:38:39 UTC 2021 Mon Jun 7 08:38:40 UTC 2021 Mon 08:38:47 UTC 2021 Mon Jun 7 08:38:48 UTC 2021 Mon Jun 7 08:38:49 UTC 2021 Mon Jun 7 08:38:56 UTC 2021 Mon Jun 7 08:38:57 UTC 2021 Mon Jun 7 08:38:58 UTC 2021 Mon Jun 7 08:39:05 UTC 2021 Mon Jun 7 08:39:06 UTC 2021 Mon Jun 7 08:39:07 UTC 2021 Mon Jun 7 08:39:14 UTC 2021 Mon Jun 7 08:39:15 UTC 2021 Mon Jun 7 08:39:16 UTC 2021 Mon Jun 7 08:39:23 UTC 2021 Mon Jun 7 08:39:24 UTC 2021 Mon Jun 7 08:39:25 UTC 2021 Mon Jun 7 08:39:32 UTC 2021 Mon Jun 7 08:39:33 UTC 2021 Mon Jun 7 08:39:34 UTC 2021 Mon Jun 7

As we can see, we connected to the pod from localhost