Troubleshooting a Broken Kubernetes Application

Your company, BeeBox, is building some applications for Kubernetes. Your developers have recently deployed an application to your cluster, but it is having some issues.

A set of Pods managed by the web-consumer deployment regularly make requests to a service that provides authentication data. Your developers are reporting that the containers are not behaving as expected.

Your task is to look at the application in question, determine the problem, and fix it.

Reading above, we need to identify what's wrong with the application and then of course fix the problem and restore communication between the pods if needed.

Identify What is Wrong with the Application

1. First lets look at the web-consumer deployment

Figure 1-1

```
cloud_user@k8s-control:~$ kubectl get deployment -n web web-consumer
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
web-consumer   2/2     2            2           12m
```

So the 2/2 pods look good and are up and running.

2. Lets get more information though by replying get > describe
3. Make sure the labels under the "Pod Template" is good
4. So those 2/2 pods we mentioned in step 1 lets take a closer look at this.

kubectl get pods -n web

Figure 1-2

```
cloud_user@k8s-control:~$ kubectl get pods -n web
NAME                            READY   STATUS    RESTARTS   AGE
web-consumer-7fb695d79c-8ms7h   1/1     Running   0          20m
web-consumer-7fb695d79c-j8dlv   1/1     Running   0          20m
```

So we see both still running. But lets still go into every pod and see more information.

5. Within that first pod, lets see more information

kubectl describe pod -n web web-consumer-7fb695d79c-8ms7h

Immediately we can see a few warning messages which are basic. So this doesn't have anything with the issues we are experiencing.

6. Lets take a look at the logs

*You can either include

kubectl logs -n web web-consumer-7fb695d79c-8ms7h

- OR –

kubectl describe pod -n web web-consumer-7fb695d79c-8ms7h -c busybox

They both spit out the same “Couldn’t resolve host ‘auth-db’”

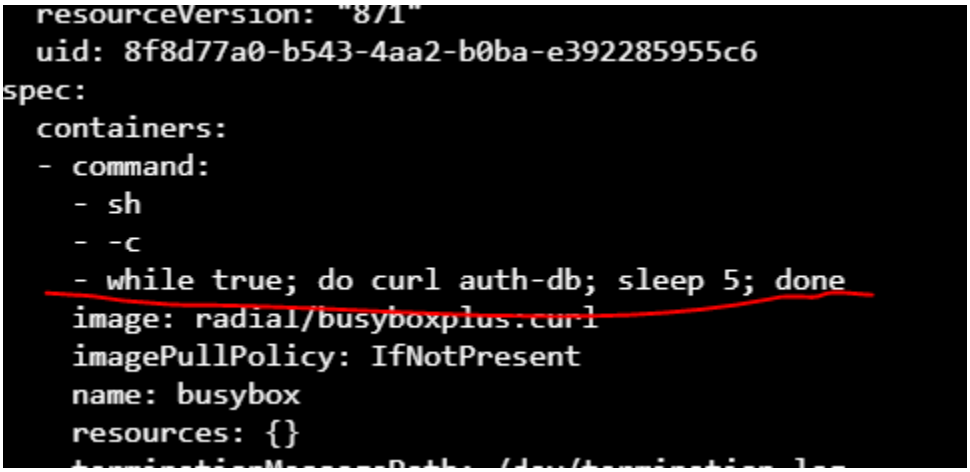
So, the problem seems the pod is not able to resolve this auth-db hostname. So more than likely DNS related

7. So lets kubectl get pod -n web <podname> -o yaml

Looking at the pod spec, we can see the “while true; do curl auth-db; sleep 5; done”

^So that means that it is doing a curl request to auth-db. And that is where we’re getting that message that says cannot resolve “auth-db” hostname.

Figure 1-3



Fix the Problem

8. Lets take a closer look at our service (auth-db)

kubectl get svc -n web auth-db

Received the error “Error from server (NotFound): services "auth-db" not found”

^So lets locate where this service is

9. Locate the auth-db hostname

Kubectl get namespace

Here we would need to go into each one until we find the auth-db

Figure 1-4

```
cloud_user@k8s-control:~$ kubectl get namespace
NAME              STATUS   AGE
data              Active   37m
default           Active   37m
kube-node-lease   Active   37m
kube-public       Active   37m
kube-system       Active   37m
web               Active   37m
cloud_user@k8s-control:~$ kubectl get svc -n data
NAME      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
auth-db   ClusterIP   10.102.69.16   <none>        80/TCP    38m
cloud_user@k8s-control:~$
```

So our pods are in the web namespace and the service is in the data namespace

*Whoever built this pod just really didn't understand how Kubernetes DNS works, and they didn't reference that service correctly from within the pod.

^ Words from the instructor

10. Lets go in and edit the deployment on the web-consumer

kubectl edit deployment -n web web-consumer

Once there, scroll down to the "spec" section where the "while true;" portion is at.

11. Replace the line with an actually FQDN

Changing the auth-db > auth-db.data.svc.cluster.local

Figure 1-5

```
spec:
  containers:
  - command:
    - sh
    - -c
    - while true; do curl auth-db.data.svc.cluster.local; sleep 5; done
    image: radial/busyboxplus:curl
    imagePullPolicy: IfNotPresent
    name: busybox
    resources: {}
```

We chose .data because remember going back to step 9 Figure 1-4. That is where this auth-db was being held at. You can see the difference from figure 1-3 to 1-5.

12. Check the pods status on that web name space

kubectl get pods -n web

Figure 1-6

We can see the old pods are terminating and the new ones are currently running. So remember even though it says up and running. We still have to make sure everything works.

13. Check the logs on one of the new pods that was created

kubectl logs -n web web-consumer-555c4f76fb-rwfv4

We can now see the nginx welcome page in the container log. Instead of an error message. Looks like the pods are able to communicate with the service now.