Using Init Containers in Kubernetes

You are working for BeeBox, a company that provides regular shipments of bees to customers. The company is in the process of deploying a shipping status application to the cluster.

The developers are building an application component designed to run in a Kubernetes pod. This component depends on a Kubernetes service called shipping-svc, and they would like their application container to delay startup when this service is not available in the cluster. Once the service becomes available, the main application container should proceed with startup.

Your task is to build a proof-of-concept showing how a pod can be designed that will delay startup of application containers until the service becomes available.

Create a Sample Pod That Uses an Init Container to Delay Startup

Since our objective is to delay the startup on the shipping-svc pod. We have to modify the pod.yml pod first.

1. Go into the pod.yml file and edit

vi pod.yml

Figure 1-1

```
apiVersion: v1
kind: Pod
metadata:
  name: shipping-web
spec:
  containers:
  - name: nginx
    image: nginx:1.19.1
  initContainers:
  - name: shipping-svc-check
    image: busybox:1.27
    command: ['sh', 'c', 'until nslookup shipping-svc; do echo waiting for shipping-svc; sleep 2; done']
~
```

Under the initContainers, we are inputting the name “shipping-svs-check” because we are checking to see if the shipping service exists.

- Under the name portion for “busybox:1.27” we need a specific version of busybox because we’re going to be doing nslookup, and some of the newer versions of busybox… nslookup has some issues. Version 1.27 is known to work.
- Commdand
    - Sh – c is doing a little bit of a shell script here in the command
    - We have it under a loop. So it will continue to run until the “nslookup shipping-svc” succeeds.
    - This means we are checking the K8s DNS to see if we can find a record for the shipping service
    - So every 2 seconds we will run the command until the service appears. And then the loop portion will exit and then the initContainer will be complete. And this will allow our main app container to procced with startup

- So basically we are delaying this startup until the shipping service (shipping-svc) becomes available

2. Remember after you completed the yml file. Use the kubectl create -f command

kubectl create -f pod.yml

^ I tried to do it with apply instead of create and it worked either way

Figure 1-2

```
cloud_user@k8s-control:~$ kubectl apply -f pod.yml
pod/shipping-web created
```

So now we have created the shipping-web pod. Doing a kubectl get pods command you can see the init status as well.

Figure 1-3

```
cloud_user@k8s-control:~$ kubectl get pods
NAME           READY   STATUS                  RESTARTS        AGE
shipping-web   0/1     Init:CrashLoopBackOff   5 (2m23s ago)   5m16s
cloud_user@k8s-control:~$ kubectl get pods
NAME           READY   STATUS                  RESTARTS        AGE
shipping-web   0/1     Init:CrashLoopBackOff   5 (2m47s ago)   5m40s
```

The reasoning behind the status not having a 0/1 may be because the shipping.svc isn't available yet.

Test Your Setup by Creating the Service and Verifying the Pod Starts Up

3. So lets create the shipping service

kubectl create -f shipping-svc.yml

Figure 1-4

```
cloud_user@k8s-control:~$ kubectl create -f shipping-svc.yml
service/shipping-svc created
pod/shipping-backend created
cloud_user@k8s-control:~$ kubectl get pods
NAME               READY   STATUS                  RESTARTS        AGE
shipping-backend   1/1     Running                 0               48s
shipping-web       0/1     Init:CrashLoopBackOff   6 (2m41s ago)   8m25s
```

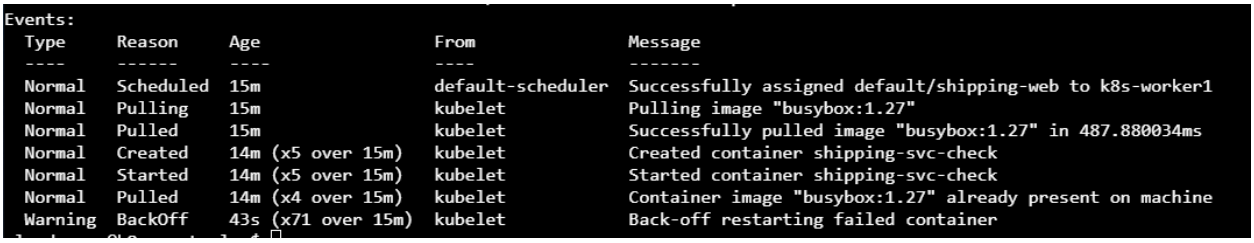
So you can see after being created. We now have a shipping-backend running. But the shipping-web is still in crashloopbackoff.

Looking into this crashloopback error may mean one or more containers are failing and restarting repeatedly. Might look into the restartPolicy.

4. First troubleshooting command I used

kubectl describe pod shipping-web

Figure 1-5



```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  15m                  default-scheduler  Successfully assigned default/shipping-web to k8s-worker1
  Normal   Pulling    15m                  kubelet            Pulling image "busybox:1.27"
  Normal   Pulled     15m                  kubelet            Successfully pulled image "busybox:1.27" in 487.880034ms
  Normal   Created    14m (x5 over 15m)    kubelet            Created container shipping-svc-check
  Normal   Started    14m (x5 over 15m)    kubelet            Started container shipping-svc-check
  Normal   Pulled     14m (x4 over 15m)    kubelet            Container image "busybox:1.27" already present on machine
  Warning  BackOff    43s (x71 over 15m)   kubelet            Back-off restarting failed container
```

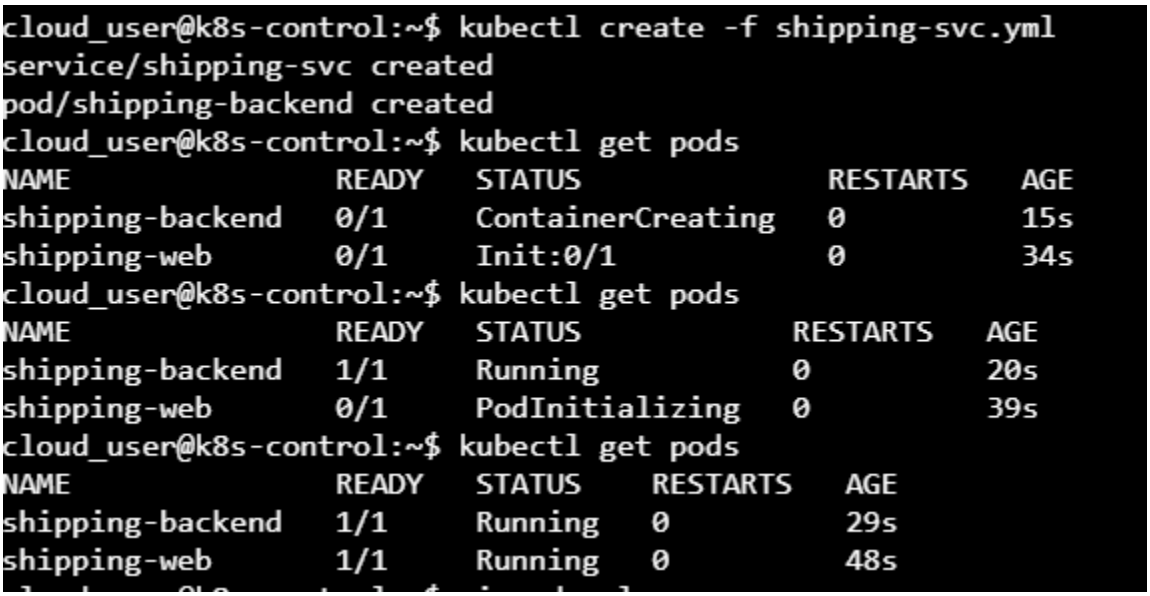
I can see one type is a “Warning” with the message “Back-off restarting failed container”

This has been restarted 8 times within 17 mins

So after doing some troubelshooting efforts it looks like the pod.yml was the issue. I rebooted the lab and had no issues.

https://sysdig.com/blog/debug-kubernetes-crashloopbackoff/

Figure 1-6



```
cloud_user@k8s-control:~$ kubectl create -f shipping-svc.yml
service/shipping-svc created
pod/shipping-backend created
cloud_user@k8s-control:~$ kubectl get pods
NAME                READY   STATUS              RESTARTS   AGE
shipping-backend    0/1     ContainerCreating   0          15s
shipping-web        0/1     Init:0/1            0          34s
cloud_user@k8s-control:~$ kubectl get pods
NAME                READY   STATUS            RESTARTS   AGE
shipping-backend    1/1     Running           0          20s
shipping-web        0/1     PodInitializing   0          39s
cloud_user@k8s-control:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
shipping-backend    1/1     Running   0          29s
shipping-web        1/1     Running   0          48s
```

So we have successfully added an init container to a pod that delays startup until a service in the cluster becomes available.