



## Exercise 3.3: Configure Probes

When large datasets need to be loaded or a complex application launched prior to client access, a `readinessProbe` can be used. The pod will not become available to the cluster until a test is met and returns a successful exit code. Both `readinessProbes` and `livenessProbes` use the same syntax and are identical other than the name. Where the `readinessProbe` is checked prior to being ready, then not again, the `livenessProbe` continues to be checked.

There are three types of liveness probes: a command returns a zero exit value, meaning success, an HTTP request returns a response code in the 200 to 399 range, and the third probe uses a TCP socket. In this example we'll use a command, `cat`, which will return a zero exit code when the file `/tmp/healthy` has been created and can be accessed.

1. Edit the YAML deployment file and add the stanza for a `readinessprobe`. Remember that when working with YAML whitespace matters. Indentation is used to parse where information should be associated within the stanza and the entire file. Do not use tabs. If you get an error about validating data, check the indentation. It can also be helpful to paste the file to this website to see how indentation affects the JSON value, which is actually what Kubernetes ingests: <https://www.json2yaml.com/> An edited file is also included in the tarball, but requires the image name to be edited to match your registry IP address.

```
student@ckad-1:~/app1$ vim simpleapp.yaml
```

YAML

simpleapp.yaml

```
1 ....
2   spec:
3     containers:
4     - image: 10.111.235.60:5000/simpleapp:latest
5       imagePullPolicy: Always
6       name: simpleapp
7       readinessProbe:           #<--This line and next five
8         periodSeconds: 5
9         exec:
10          command:
11            - cat
12            - /tmp/healthy
13       resources: {}
14 ....
```

2. Delete and recreate the `try1` deployment.

```
student@ckad-1:~/app1$ kubectl delete deployment try1
```

```
1 deployment.apps "try1" deleted
```

```
student@ckad-1:~/app1$ kubectl create -f simpleapp.yaml
```

```
1 deployment.apps/try1 created
```

3. The new `try1` deployment should reference six pods, but show zero available. They are all missing the `/tmp/healthy` file.

```
student@ckad-1:~/app1$ kubectl get deployment
```

```

1 NAME      READY    UP-TO-DATE    AVAILABLE    AGE
2 nginx     1/1      1             1            19m
3 registry  1/1      1             1            19m
4 try1      0/6      6             0            15s

```

4. Take a closer look at the pods. Choose one of the `try1` pods as a test to create the health check file.

```
student@ckad-1:~/app1$ kubectl get pods
```

```

1 NAME                                READY    STATUS    RESTARTS    AGE
2 nginx-6b58d9cdfd-g7lnk             1/1      Running   1            40m
3 registry-795c6c8b8f-7vwdn          1/1      Running   1            40m
4 try1-9869bdb88-2wfnr                0/1      Running   0            26s
5 try1-9869bdb88-6bkn1                0/1      Running   0            26s
6 try1-9869bdb88-786v8                0/1      Running   0            26s
7 try1-9869bdb88-gmvs4                0/1      Running   0            26s
8 try1-9869bdb88-lfv1x                0/1      Running   0            26s
9 try1-9869bdb88-rtchc                0/1      Running   0            26s

```

5. Run the bash shell interactively and touch the `/tmp/healthy` file.

```
student@ckad-1:~/app1$ kubectl exec -it try1-9869bdb88-rtchc -- /bin/bash
```

```
root@try1-9869bdb88-rtchc:/# touch /tmp/healthy
```

```
root@try1-9869bdb88-rtchc:/# exit
```

```
1 exit
```

6. Wait at least five seconds, then check the pods again. Once the probe runs again the container should show available quickly. The pod with the existing `/tmp/healthy` file should be running and show 1/1 in a READY state. The rest will continue to show 0/1.

```
student@ckad-1:~/app1$ kubectl get pods
```

```

1 NAME                                READY    STATUS    RESTARTS    AGE
2 nginx-6b58d9cdfd-g7lnk             1/1      Running   1            44m
3 registry-795c6c8b8f-7vwdn          1/1      Running   1            44m
4 try1-9869bdb88-2wfnr                0/1      Running   0            4m
5 try1-9869bdb88-6bkn1                0/1      Running   0            4m
6 try1-9869bdb88-786v8                0/1      Running   0            4m
7 try1-9869bdb88-gmvs4                0/1      Running   0            4m
8 try1-9869bdb88-lfv1x                0/1      Running   0            4m
9 try1-9869bdb88-rtchc                1/1      Running   0            4m

```

7. Touch the file in the remaining pods. Consider using a **for** loop, as an easy method to update each pod. Note the `>` shown in the output represents the secondary prompt, you would not type in that character

```

student@ckad-1:~$ for name in try1-9869bdb88-2wfnr try1-9869bdb88-6bkn1 \
> try1-9869bdb88-786v8 try1-9869bdb88-gmvs4 try1-9869bdb88-lfv1x
> do
> kubectl exec $name -- touch /tmp/healthy
> done

```

8. It may take a short while for the probes to check for the file and the health checks to succeed.

```
student@ckad-1:~/app1$ kubectl get pods
```

	NAME	READY	STATUS	RESTARTS	AGE
1					
2	nginx-6b58d9cdfd-g7lnk	1/1	Running	1	1h
3	registry-795c6c8b8f-7vwdn	1/1	Running	1	1h
4	try1-9869bdb88-2wfnr	1/1	Running	0	22m
5	try1-9869bdb88-6bknl	1/1	Running	0	22m
6	try1-9869bdb88-786v8	1/1	Running	0	22m
7	try1-9869bdb88-gmvs4	1/1	Running	0	22m
8	try1-9869bdb88-lfvlx	1/1	Running	0	22m
9	try1-9869bdb88-rtchc	1/1	Running	0	22m

9. Now that we know when a pod is healthy, we may want to keep track that it stays healthy, using a `livenessProbe`. You could use one probe to determine when a pod becomes available and a second probe, to a different location, to ensure ongoing health.

Edit the deployment again. Add in a `livenessProbe` section as seen below. This time we will add a `Sidecar` container to the pod running a simple application which will respond to port 8080. Note that the dash (-) in front of the name. Also `goproxy` is indented the same number of spaces as the - in front of the `image:` line for `simpleapp` earlier in the file. In this example that would be seven spaces

```
student@ckad-1:~/app1$ vim simpleapp.yaml
```

YA  
ML

simpleapp.yaml

```
1  ....
2      terminationMessagePath: /dev/termination-log
3      terminationMessagePolicy: File
4      - name: goproxy                #<-- Indented 7 spaces, add lines from here...
5        image: k8s.gcr.io/goproxy:0.1
6        ports:
7          - containerPort: 8080
8        readinessProbe:
9          tcpSocket:
10             port: 8080
11             initialDelaySeconds: 5
12             periodSeconds: 10
13        livenessProbe:                #<-- This line is 9 spaces indented, fyi
14          tcpSocket:
15            port: 8080
16            initialDelaySeconds: 15
17            periodSeconds: 20        #<-- ....to here
18        dnsPolicy: ClusterFirst
19        restartPolicy: Always
20  ....
```

10. Delete and recreate the deployment.

```
student@ckad-1:~$ kubectl delete deployment try1
```

```
1 deployment.apps "try1" deleted
```

```
student@ckad-1:~$ kubectl create -f simpleapp.yaml
```

```
1 deployment.apps/try1 created
```

11. View the newly created pods. You'll note that there are two containers per pod, and only one is running. The new `simpleapp` containers will not have the `/tmp/healthy` file, so they will not become available until we touch the `/tmp/healthy` file again. We could include a command which creates the file into the container arguments. The output below shows it can take a bit for the old pods to terminate.

```
student@ckad-1:~$ kubectl get pods
```

	NAME	READY	STATUS	RESTARTS	AGE
1	nginx-6b58d9cdfd-g7lnk	1/1	Running	1	13h
2	registry-795c6c8b8f-7vwdn	1/1	Running	1	13h
3	try1-76cc5ffcc6-4rjvh	1/2	Running	0	3s
4	try1-76cc5ffcc6-bk5f5	1/2	Running	0	3s
5	try1-76cc5ffcc6-d8n5q	0/2	ContainerCreating	0	3s
6	try1-76cc5ffcc6-mm6tw	1/2	Running	0	3s
7	try1-76cc5ffcc6-r9q5n	1/2	Running	0	3s
8	try1-76cc5ffcc6-tx4dz	1/2	Running	0	3s
9	try1-9869bdb88-2wfnr	1/1	Terminating	0	12h
10	try1-9869bdb88-6bkn1	1/1	Terminating	0	12h
11	try1-9869bdb88-786v8	1/1	Terminating	0	12h
12	try1-9869bdb88-gmvs4	1/1	Terminating	0	12h
13	try1-9869bdb88-lfvlx	1/1	Terminating	0	12h
14	try1-9869bdb88-rtchc	1/1	Terminating	0	12h
15					

12. Create the health check file for the readinessProbe. You can use a **for** loop again for each action, this setup will leverage labels so you don't have to look up the pod names. As there are now two containers in the pod, you should include the container name for which one will execute the command. If no name is given, it will default to the first container. Depending on how you edited the YAML file try1 should be the first pod and goproxy the second. To ensure the correct container is updated, add **-c simpleapp** to the **kubectl** command. Your pod names will be different. Use the names of the newly started containers from the **kubectl get pods** command output. Note the **>** character represents the secondary prompt, you would not type in that character.

```
student@ckad-1:~$ for name in $(kubectl get pod -l app=try1 -o name)
> do
> kubectl exec $name -c simpleapp -- touch /tmp/healthy
> done
```

13. In the next minute or so the Sidecar container in each pod, which was not running, will change status to Running. Each should show 2/2 containers running.

```
student@ckad-1:~$ kubectl get pods
```

	NAME	READY	STATUS	RESTARTS	AGE
1	nginx-6b58d9cdfd-g7lnk	1/1	Running	1	13h
2	registry-795c6c8b8f-7vwdn	1/1	Running	1	13h
3	try1-76cc5ffcc6-4rjvh	2/2	Running	0	3s
4	try1-76cc5ffcc6-bk5f5	2/2	Running	0	3s
5	try1-76cc5ffcc6-d8n5q	2/2	Running	0	3s
6	try1-76cc5ffcc6-mm6tw	2/2	Running	0	3s
7	try1-76cc5ffcc6-r9q5n	2/2	Running	0	3s
8	try1-76cc5ffcc6-tx4dz	2/2	Running	0	3s
9					

14. View the events for a particular pod. Even though both containers are currently running and the pod is in good shape, note the events section shows the issue.

```
student@ckad-1:~/app1$ kubectl describe pod try1-76cc5ffcc6-tx4dz | tail
```

1	Normal	SuccessfulMountVolume	9m	kubelet, ckad-1-lab-x6dj
2		MountVolume.SetUp succeeded for volume "default-token-jf69w"		
3	Normal	Pulling	9m	kubelet, ckad-1-lab-x6dj
4		pulling image "10.108.143.90:5000/simpleapp"		
5	Normal	Pulled	9m	kubelet, ckad-1-lab-x6dj
6		Successfully pulled image "10.108.143.90:5000/simpleapp"		
7	Normal	Created	9m	kubelet, ckad-1-lab-x6dj
8		Created container		
9	Normal	Started	9m	kubelet, ckad-1-lab-x6dj
10		Started container		
11	Normal	Pulling	9m	kubelet, ckad-1-lab-x6dj

```

12 pulling image "k8s.gcr.io/goproxy:0.1"
13   Normal    Pulled                9m                kubelet, ckad-1-lab-x6dj
14 Successfully pulled image "k8s.gcr.io/goproxy:0.1"
15   Normal    Created                  9m                kubelet, ckad-1-lab-x6dj
16 Created container
17   Normal    Started                  9m                kubelet, ckad-1-lab-x6dj
18 Started container
19   Warning   Unhealthy                4m (x60 over 9m)  kubelet, ckad-1-lab-x6dj
20 Readiness probe failed: cat: /tmp/healthy: No such file or directory

```

15. If you look for the status of each container in the pod, they should show that both are Running and ready showing True.

```
student@ckad-1:~/app1$ kubectl describe pod try1-76cc5ffcc6-tx4dz | grep -E 'State|Ready'
```

```

1     State:      Running
2     Ready:      True
3     State:      Running
4     Ready:      True
5 Ready        True
6 ContainersReady  True

```