**Health Checks**

Kubernetes health checks by example

In order to verify if a container in a pod is healthy and ready to serve traffic, Kubernetes provides for a range of health checking mechanisms. Health checks, or **probes** as they are called in Kubernetes, are carried out by the [kubelet](https://kubernetes.io/docs/admin/kubelet/) to determine when to restart a container (for livenessProbe) and used by services and deployments to determine if a pod should receive traffic (for readinessProbe).

We will focus on HTTP health checks in the following. Note that it is the responsibility of the application developer to expose a URL that the kubelet can use to determine if the container is healthy (and potentially ready).

Let’s create a [pod](https://github.com/openshift-evangelists/kbe/blob/master/specs/healthz/pod.yaml) that exposes an endpoint /health, responding with a HTTP 200 status code:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/healthz/pod.yaml

In the pod specification we’ve defined the following:

livenessProbe:

initialDelaySeconds: 2

periodSeconds: 5

httpGet:

path: /health

port: 9876

Above means that Kubernetes will start checking the /health endpoint, after initially waiting 2 seconds, every 5 seconds.

If we now look at the pod we can see that it is considered healthy:

$ kubectl describe pod hc

Name: hc

Namespace: default

Security Policy: anyuid

Node: 192.168.99.100/192.168.99.100

Start Time: Tue, 25 Apr 2017 16:21:11 +0100

Labels: <none>

Status: Running

...

Events:

FirstSeen LastSeen Count From SubobjectPath Type Reason Message

--------- -------- ----- ---- ------------- -------- ------ -------

3s 3s 1 {default-scheduler } Normal Scheduled Successfully assigned hc to 192.168.99.100

3s 3s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Pulled Container image "mhausenblas/simpleservice:0.5.0"

already present on machine

3s 3s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Created Created container with docker id 8a628578d6ad; Security:[seccomp=unconfined]

2s 2s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Started Started container with docker id 8a628578d6ad

Now we launch a [bad pod](https://github.com/openshift-evangelists/kbe/blob/master/specs/healthz/badpod.yaml), that is, a pod that has a container that randomly (in the time range 1 to 4 sec) does not return a 200 code:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/healthz/badpod.yaml

Looking at the events of the bad pod, we can see that the health check failed:

$ kubectl describe pod badpod

...

Events:

FirstSeen LastSeen Count From SubobjectPath Type Reason Message

--------- -------- ----- ---- ------------- -------- ------ -------

1m 1m 1 {default-scheduler } Normal Scheduled Successfully assigned badpod to 192.168.99.100

1m 1m 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Created Created container with docker id 7dd660f04945; Security:[seccomp=unconfined]

1m 1m 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Started Started container with docker id 7dd660f04945

1m 23s 2 {kubelet 192.168.99.100} spec.containers{sise} Normal Pulled Container image "mhausenblas/simpleservice:0.5.0" already present on machine

23s 23s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Killing Killing container with docker id 7dd660f04945: pod "badpod\_default(53e5c06a-29cb-11e7-b44f-be3e8f4350ff)" container "sise" is unhealthy, it will be killed and re-created.

23s 23s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Created Created container with docker id ec63dc3edfaa; Security:[seccomp=unconfined]

23s 23s 1 {kubelet 192.168.99.100} spec.containers{sise} Normal Started Started container with docker id ec63dc3edfaa

1m 18s 4 {kubelet 192.168.99.100} spec.containers{sise} Warning Unhealthy Liveness probe failed: Get http://172.17.0.4:9876/health: net/http: request canceled (Client.Timeout exceeded while awaiting headers)

This can also be verified as follows:

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

badpod 1/1 Running 4 2m

hc 1/1 Running 0 6m

From above you can see that the badpod had already been re-launched 4 times, since the health check failed.

In addition to a livenessProbe you can also specify a readinessProbe, which can be configured in the same way but has a different use case and semantics: it’s used to check the start-up phase of a container in the pod. Imagine a container that loads some data from external storage such as S3 or a database that needs to initialize some tables. In this case you want to signal when the container is ready to serve traffic.

Let’s create a [pod](https://github.com/openshift-evangelists/kbe/blob/master/specs/healthz/ready.yaml) with a readinessProbe that kicks in after 10 seconds:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/healthz/ready.yaml

Looking at the events of the pod, we can see that, eventually, the pod is ready to serve traffic:

$ kubectl describe pod ready

...

Conditions: [0/1888]

Type Status

Initialized True

Ready True

PodScheduled True

...

You can remove all the created pods with:

$ kubectl delete pod/hc pod/ready pod/badpod

Learn more about configuring probes, including TCP and command probes, via the [docs](https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/)