Kubernetes健康检查 & Pod状态

健康检查

健康检查的类型

1. Liveness probe

Kubelet使用liveness probe（存活探针）来确定何时重启容器。例如，当应用程序处于运行状态但无法做进一步操作，liveness探针将捕获到deadlock，重启处于该状态下的容器，使应用程序在存在bug的情况下依然能够继续运行下去（谁的程序还没几个bug呢）。

- 探测方式
  - 执行命令

    如果命令执行成功，将返回0，kubelet就会认为该容器是活着的并且很健康。如果返回非0值，kubelet就会杀掉这个容器并重启它。
  - HTTP请求

    任何大于200小于400的返回码都会认定是成功的返回码。其他返回码都会被认为是失败的返回码。
  - TCP探针

    kubelet将尝试在指定端口上打开容器的套接字。 如果可以建立连接，容器被认为是健康的，如果不能就认为是失败的。
- 例子: 点击

2. Readness probe

Kubelet使用readiness probe（就绪探针）来确定容器是否已经就绪可以接受流量。只有当Pod中的容器都处于就绪状态时kubelet才会认定该Pod处于就绪状态。该信号的作用是控制哪些Pod应该作为service的后端。如果Pod处于非就绪状态，那么它们将会被从service的load balancer中移除。

- 探测方式
  - 与liveness probe相同

liveness & readness 区别

LivenessProbe探针： 用于判断容器是否存活，即Pod是否为running状态，如果LivenessProbe探针探测到容器不健康，则kubelet将kill掉容器，并根据容器的重启策略是否重启，如果一个容器不包含LivenessProbe探针，则Kubelet认为容器的LivenessProbe探针的返回值永远成功。 ReadinessProbe探针： 用于判断容器是否启动完成，即容器的Ready是否为True，可以接收请求，如果ReadinessProbe探测失败，则容器的Ready将为False，控制器将此Pod的Endpoint从对应的service的Endpoint列表中移除，从此不再将任何请求调度此Pod上，直到下次探测成功。

使用建议

1. 建议对全部服务同时设置服务（readiness）和Container（liveness）的健康检查
2. 通过TCP对端口检查形式（TCPSocketAction），仅适用于端口已关闭或进程停止情况。因为即使服务异常，只要端口是打开状态，健康检查仍然是通过的。
3. 基于第二点，一般建议用ExecAction自定义健康检查逻辑，或采用HTTP Get请求进行检查（HTTPGetAction）。
4. 无论采用哪种类型的探针，一般建议设置检查服务（readiness）的时间短于检查Container（liveness）的时间，也可以将检查服务（readiness）的探针与Container（liveness）的探针设置为一致。目的是故障服务先下线，如果过一段时间还无法自动恢复，那么根据重启策略，重启该container、或其他机器重新创建一个pod恢复故障服务。

Probe配置

通用配置

initialDelaySeconds：容器启动后第一次执行探测是需要等待多少秒。

periodSeconds：执行探测的频率。默认是10秒，最小1秒。

timeoutSeconds：探测超时时间。默认1秒，最小1秒。

successThreshold：探测失败后，最少连续探测成功多少次才被认定为成功。默认是1。对于liveness必须是1。最小值是1。

failureThreshold：探测成功后，最少连续探测失败多少次才被认定为失败。默认是3。最小值是1。

httpGet配置

host：连接的主机名，默认连接到pod的IP。你可能想在http header中设置“Host”而不是使用IP。

scheme：连接使用的schema，默认HTTP。

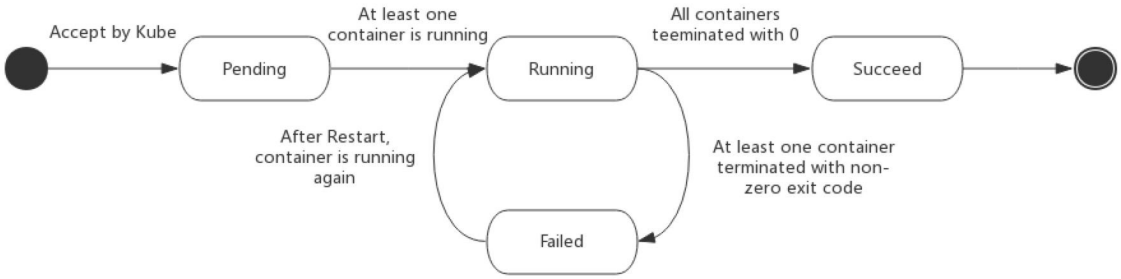
path: 访问的HTTP server的path。

httpHeaders：自定义请求的header。HTTP运行重复的header。

port：访问的容器的端口名字或者端口号。端口号必须介于1和65535之间。

Pod状态

Pod Phase



挂起（Pending）：Pod 已被 Kubernetes 系统接受，但有一个或者多个容器镜像尚未创建。等待时间包括调度 Pod 的时间和通过网络下载镜像的时间，这可能需要花点时间。

运行中（Running）：该 Pod 已经绑定到了一个节点上，Pod 中所有的容器都已被创建。至少有一个容器正在运行，或者正处于启动或重启状态。

成功（Succeeded）：Pod 中的所有容器都被成功终止，并且不会再重启。

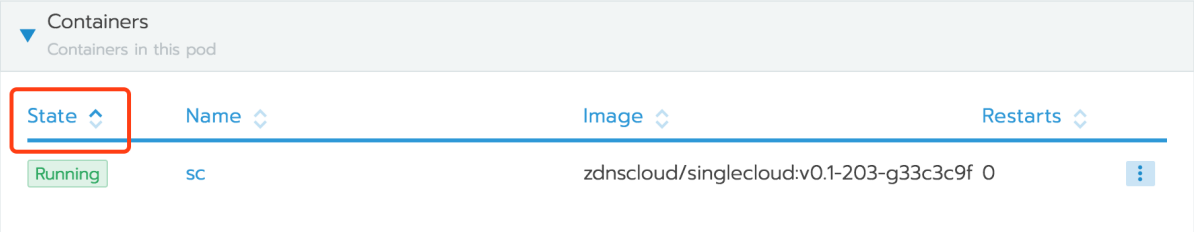
失败（Failed）：Pod 中的所有容器都已终止了，并且至少有一个容器是因为失败终止。也就是说，容器以非0状态退出或者被系统终止。

未知（Unknown）：因为某些原因无法取得 Pod 的状态，通常是因为与 Pod 所在主机通信失败。

使用Kubectl获取Pod Phase, 此处被称为STATUS

```
tanxu@zdns-PowerEdge-R720:~$ kubectl get pods --all-namespaces
NAMESPACE           NAME                                                     READY   STATUS    RESTARTS   AGE
cattle-prometheus   exporter-kube-state-cluster-monitoring-58f946d4d7-d6s2s   1/1     Running   0          4h50m
cattle-prometheus   exporter-node-cluster-monitoring-ghblf                   1/1     Running   0          4h50m
cattle-prometheus   exporter-node-cluster-monitoring-qzzst                   1/1     Running   0          4h50m
cattle-prometheus   exporter-node-cluster-monitoring-zccpj                   1/1     Running   0          4h50m
cattle-prometheus   grafana-cluster-monitoring-65d7cfccd8-c75fh              2/2     Running   0          4h50m
cattle-prometheus   prometheus-cluster-monitoring-0                          5/5     Running   1          4h50m
```

在Rancher中获取Pod Phase, 此处被称为State



Pod Status对象

包含一个 PodCondition 数组

PodCondition 数组的每个元素都有一个 type 字段和一个 status 字段。

type 字段是字符串，可能的值有

- PodScheduled
- Ready
- Initialized
- Unschedulable

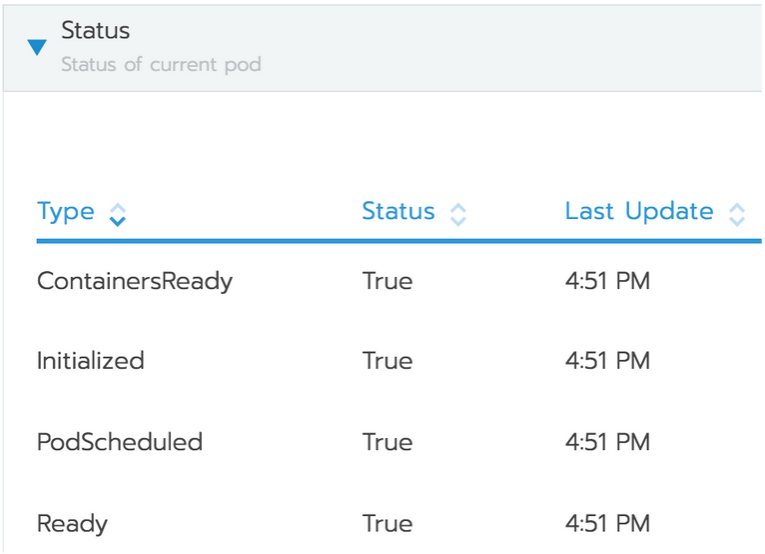
status 字段是一个字符串，可能的值有

- True
- False
- Unknown

使用kubectl查看PodCondition

```
status:
  conditions:
  - lastProbeTime: null
    lastTransitionTime: "2019-04-25T07:51:57Z"
    status: "True"
    type: Initialized
  - lastProbeTime: null
    lastTransitionTime: "2019-04-25T07:52:15Z"
    status: "True"
    type: Ready
  - lastProbeTime: null
    lastTransitionTime: "2019-04-25T07:52:15Z"
    status: "True"
    type: ContainersReady
  - lastProbeTime: null
    lastTransitionTime: "2019-04-25T07:51:57Z"
    status: "True"
    type: PodScheduled
```

使用rancher查看PodCondition



Reference

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/

https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/

https://jimmysong.io/kubernetes-handbook/guide/configure-liveness-readiness-probes.html

https://blog.51cto.com/newfly/2137136

https://juejin.im/post/5990e5f56fb9a03c4300a891

http://dockone.io/article/2587