前言

在上一章，我们学习了 `k8s` 如何处理 `Pod` 的滚动发布，滚动发布的主要目的是做到零宕机完成环境更新。

那么问题来了， `kubernetes` 到底是以什么依据，判断我们 `Pod` 启动成功的?

什么是健康度检查?

我们在之前的部署知道，当 `Pod` 的状态为 `Running` 时，该 `Pod` 就可以被分配流量（可以访问到）了。**但是，这种检查方式对于一部分Pod来说是不靠谱的。**

有写过后端的同学可能了解，一般一个后端容器启动成功，不一定不代表服务启动成功。在后端容器启动后，部分 MySQL，消息队列，配置文件等其他服务的连接还在初始化，但是容器的外部状态却是启动成功。在这种情况下，直接去访问 Pod 必然会有问题。

那么有没有什么办法可以自己控制流量分配的标准呢？这就要提到我们下面要写到的概念 —— 服务探针

什么是服务探针?

探针一词，和古代银针试毒的概念差不多 —— 将银针放入水中，如果银针变黑，则代表有毒；如果没有变黑，则代表正常。

那么在 `kubernetes` 中，探针用来检测 `Pod` 可用情况的。在 `kubernetes` 中，有三种探针可以使用：

1. 存活探针 LivenessProbe

第一种是存活探针。存活探针是对运行中的容器检测的。如果想检测你的服务在运行中有没有发生崩溃，服务有没有中途退出或无响应，可以使用这个探针。

2. 可用探针 ReadinessProbe

第二种是可用探针。作用是用来检测 Pod 是否允许被访问到（是否准备好接受流量）。如果你的服务加载很多数据，或者有其他需求要求在特定情况下不被分配到流量，那么可以用这个探针。

如果探针检测失败，流量就不会分配给该 Pod。在没有配置该探针的情况下，会一直将流量分配给 Pod。**当然，探针检测失败，Pod 不会被杀死。**

3. 启动探针 StartupProbe

第三种是启动探针。作用是用来检测 Pod 是否已经启动成功。如果你的服务启动需要一些加载时长（例如初始化日志，等待其他调用的服务启动成功）才代表服务启动成功，则可以用这个探针。

如果探针检测失败，该 Pod 就会被杀死重启。在没有配置该探针的情况下，默认不会杀死 `Pod` 。在启动探针运行时，其他所有的探针检测都会失效。

总结

`Kubernetes` 里面内置了三种健康度探针，可以分别在启动时和运行时为我们的 Pod 做检测。下面是一个对比表格：

探针名称	在哪个环节触发	作用	检测失败对Pod的反应
存活探针	Pod 运行时	检测服务是否崩溃，是否需要重启服务	杀死 Pod 并重启
可用探针	Pod 运行时	检测服务是不是允许被访问到。	停止Pod的访问调度，不会被杀死重启
启动探针	Pod 运行时	检测服务是否启动成功	杀死 Pod 并重启

当然，配置的方式也很简单。我们只需要在 `containers.livenessProbe/readinessProbe/StartupProbe` 下配置即可

三种方式探测方式

检测方式：

1. ExecAction

这种方式是通过在 Pod 的容器内执行预定的 Shell 脚本命令。如果执行的命令没有报错退出（返回值为0），代表容器状态健康。否则就是有问题的。

我们以下面为例：这是一个创建 Pod 的配置文件模版。可以看到，里面配置了一个存活探针 LivenessProbe + ExecAction 命令检测。其中， `livenessProbe.exec` 代表去执行一段命令， `command` 则是要执行的探针命令。 `livenessProbe` 代表声明一个存活探针。

yaml 复制代码

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: registry.aliyuncs.com/google_containers/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```

当我们的 Pod 启动成功时，会自动执行下面的这个命令：新建一个 `/tmp/healthy` 文件 => 睡眠 30 秒 => 删除 `/tmp/healthy` 文件 => 睡眠 600 秒。

shell 复制代码

```shell
touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
```

而我们的探针检测，在第一次探测等待5秒后，会去尝试访问 `/tmp/healthy` 文件来判断检测结果。然而，只有在 `Pod` 运行的前30秒，这个文件才存在。在第30秒后，文件被

我们创建一下这个 Pod ，然后来看下效果：

shell 复制代码

```
vim ./liveness-exec.yaml
kubectl apply -f ./liveness-exec.yaml && date && kubectl get pods | grep liveness-exec
```

```
[root@master pod]# date && kubectl get pods | grep liveness-exec
Sun Nov 29 12:48:26 CST 2020
liveness-exec                1/1     Running   0          10s
```


等待30秒后，我们通过 kubectl describe 命令看下 Pod 的运行全览状态：

shell 复制代码

```
kubectl describe pods liveness-exec
```

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  2m34s                default-scheduler  Successfully assigned default/liveness-exec to node1
  Normal   Pulled     6m18s                kubelet            Successfully pulled image "registry.aliyuncs.com/google_co
ntainers/busybox" in 612.636963ms
  Normal   Pulled     5m4s                 kubelet            Successfully pulled image "registry.aliyuncs.com/google_co
ntainers/busybox" in 590.70006ms
  Warning  Unhealthy  4m20s (x6 over 5m45s)  kubelet          Liveness probe failed: cat: can't open '/tmp/healthy': No
such file or directory
  Normal   Killing    4m20s (x2 over 5m35s)  kubelet          Container liveness failed liveness probe, will be restarte
d
```


可以看到，在运行一段时间后，探针被检测失败，Pod 被迫重启，接着创建了新的 Pod。

2. TCPSocketAction

这种方式是使用 TCP 套接字检测。 Kubernetes 会尝试在 Pod 内与指定的端口进行连接。如果能建立连接（Pod的端口打开了），这个容器就代表是健康的，如果不能，则代表这个 Pod 就是有问题的。

下面这个 Pod 配置文件就是个例子：这里定义了一个可用探针 + 存活探针，检测方式为TCP检测。探针会在容器启动成功5秒后开始检测，每10秒检测一次，每次会尝试访问 Pod 的8080端口。

yaml 复制代码

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: tcp-socket-action
  labels:
    app: tcp-socket-action
```

```
    image: registry.cn-hangzhou.aliyuncs.com/janlay/goproxy:0.1
    ports:
    - containerPort: 8080
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
    livenessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 20
```

**

3. HTTPGetAction

这种方式是使用 HTTP GET 请求。Kubernetes 会尝试访问 Pod 内指定的API路径。如果返回200，代表容器就是健康的。如果不能，代表这个 Pod 是有问题的。

这里我们配置了一个存活探针。探针将会在容器启动成功后3秒钟开始进行检测，每隔3秒检测一次。每次都会携带 `httpHeaders` 内填写的请求头，并尝试访问 `8080` 端口下的 `/healthz` 地址。

yaml 复制代码

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: registry.cn-hangzhou.aliyuncs.com/janlay/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
```

而容器内 `/healthz` 地址的编写规则是：容器启动 `10` 秒钟内会返回 `200` 请求码。之后统一返回 `500` 状态码

go 复制代码

```go
http.HandleFunc("/healthz", func(w http.ResponseWriter, r *http.Request) {
    duration := time.Now().Sub(started)
    if duration.Seconds() > 10 {
        w.WriteHeader(500)
        w.Write([]byte(fmt.Sprintf("error: %v", duration.Seconds())))
    } else {
        w.WriteHeader(200)
        w.Write([]byte("ok"))
    }
})
```

接着我们保存配置文件，等10-20秒钟后并看下结果

shell 复制代码

```shell
vim ./liveness-http.yaml
kubectl apply -f ./liveness-http.yaml
kubectl describe pods liveness-http # 等10-20秒后在执行
```

可以看到，Pod 启动一段时间后，探针检测到返回值500后，标记检测失败，并回收了 Pod 重新创建。

```
Events:
  Type     Reason     Age                   From               Message
  ----     ------     ----                  ----               -------
  Normal   Scheduled  2m6s                  default-scheduler  Successfully assigned default/liveness-http to node1
  Normal   Pulled     5m48s                 kubelet            Successfully pulled image "registry.cn-hangzhou.aliyuncs.com/janlay/liveness" in 1.896919266s
  Normal   Pulled     5m29s                 kubelet            Successfully pulled image "registry.cn-hangzhou.aliyuncs.com/janlay/liveness" in 619.986689ms
  Normal   Started    5m11s (x3 over 5m48s) kubelet            Started container liveness
  Normal   Pulled     5m11s                 kubelet            Successfully pulled image "registry.cn-hangzhou.aliyuncs.com/janlay/liveness" in 601.858147ms
  Normal   Created    5m11s (x3 over 5m48s) kubelet            Created container liveness
  Warning  Unhealthy  4m54s (x9 over 5m36s) kubelet            Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing    4m54s (x3 over 5m30s) kubelet            Container liveness failed liveness probe, will be restarted
  Normal   Pulling    4m54s (x4 over 5m50s) kubelet            Pulling image "registry.cn-hangzhou.aliyuncs.com/janlay/liveness"
```


控制探针检测的行为

上面的部分都是如何对Pod进行检测，那我们有没有什么参数可以修改检测行为呢？`Kubenetes` 给我们准备了一些额外的参数帮助我们来定义检测行为：

- initialDelaySeconds：容器初始化等待多少秒后才会触发探针。默认为0秒。
- periodSeconds：执行探测的时间间隔。默认10秒，最少1秒。
- timeoutSeconds：探测超时时间。默认1秒，最少1秒。
- successThreshold：探测失败后的最小连续成功数量。默认是1。
- failureThreshold：探测失败后的重试次数。默认是3次，最小是1次。

kubernetes.io/zh/docs/tas...

上一章 下一章

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论