30.就绪探针

我们知道，如果Pod的标签与服务的pod标签选择器匹配，pod就会加入到服务的Endpoints列表中，作为该服务的后端pod。对于新创建的pod，如果带有与服务相匹配标签，它就会成为服务的一部分，外部请求就可能马上流入到该pod上。但这样可能就会有问题，如果这个pod没有准备好立即接受请求呢?

Pod本身可能需要时间加载配置或者数据，或者需要执行一个预热的过程以防止第一个到达的用户请求花费太长时间而对用户体验造成的影响。在这种情况下，我们就不希望pod立即就开始接收外部请求，特别是当已经运行的其他pod实例能够很好地处理请求时。所以不要将请求转发给还处于启动过程中的pod，除非它完全准备妥当。

什么是就绪探针

就绪探针英文为Readiness Probe。在之前的章节中我们学习过存活探针以及它如何通过确保异常容器自动重启来保持应用程序的正常运行。与存活探针一样，Kubernetes允许我们为pod定义一个就绪探针。

通过定期地调用就绪探针来判断指定的pod是否应该接收客户端的请求。如果容器的就绪探针返回成功，那么就意味着容器准备好接收外部请求了。

这个准备就绪的概念显然特定于每个容器的。Kubernetes仅仅检查运行在容器中的应用程序是否会响应一个简单的GET/请求，或者通过一个特定的URL地址让应用程序执行一系列的检查以决定是否就绪。对于这种涉及到应用程序相关细节的就绪探针，就是开发人员在开发时需要考虑的了。

就绪探针的类型

与存活探针一样，就绪探针也有三种类型：

- HTTP GET探针，向容器发送HTTP GET请求，通过响应的HTTP状态码判断容器是否准备就绪。
- TCP Soket探针，尝试与容器指定端口建立TCP连接。连接一旦成功建立，则判定容器为准备就绪状态。
- Exec探针，在容器中执行任意的命令，然后检查命令的退出状态码来决定容器的状态。

就绪探针原理

当启动容器的时候，可以为Kubernetes配置一个等待时间，经过等待时间后才可以执行第一次就绪检查。之后，它会周期性地调用探针，并根据就绪探针的结果采取相应的动作。如果某个pod报告它尚未准备就绪，就会从服务的Endpoints中移除该pod；如果pod准备就绪，它就会被重新添加到服务的Endpoints中。

与存活探针不同，如果一个容器就绪状态检查失败，容器不会被终止或者重启。这是存活探针和就绪探针之间很重要的一个区别。存活探针通过终止并替换异常的容器来保持pod的正常工作，而就绪探针则确保只有准备就绪的pod才可以接收请求。这在容器启动时尤为必要，当然在容器运行一段时间后也是有用的。

为什么需要就绪探针

假定有一组pod（如运行应用服务器的pod）依赖某个服务，该服务由另一个pod（如后端数据库pod）提供。如果某个前端pod遇到连接问题并且无法再访问数据库，那么就绪探针就会通知Kubernetes该pod还没有准备好接收请求，从而避免用户请求达到该pod上。如果其他pod实例没有遇到这种类型的连接问题且运行正常，它们就可以正常处理请求。就绪探针确保客户端只与健康的pod通信，用户甚至都感知不到系统出了问题。

为pod指定就绪探针

现在我们来修改之前创建的名为test-tomcat-deploy的deployment资源，为其添加一个就绪探针。

可以使用kubectl edit命令将探针添加到该deployment的pod模板中：

kubectl edit deployment test-tomcat-deploy

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "2"
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"name":"test-tomcat
hLabels":{"app":"tomcat"}},"template":{"metadata":{"labels":{"app":"tomcat"}},"spec":{"containers"
ainerPort":8080,"name":"http"}]}]}}}}
  creationTimestamp: "2021-01-12T03:03:54Z"
  generation: 2
  name: test-tomcat-deploy
  namespace: default
  resourceVersion: "1193224"
  selfLink: /apis/apps/v1/namespaces/default/deployments/test-tomcat-deploy
  uid: 255425b6-bf36-4f07-ae0d-1dc2706483c9
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: tomcat
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: tomcat
    spec:
      containers:
      - image: tomcat:8.5.34-jre8-alpine
        imagePullPolicy: IfNotPresent
        name: tomcat
        ports:
        - containerPort: 8080
          name: http
          protocol: TCP
        readinessProbe:
          exec:
            command:
            - ls
            - /var/ready
          failureThreshold: 3
```

可以为pod中的每个容器添加一个readinessProbe

```
[root@k8s-master test]# kubectl edit deployment test-tomcat-deploy
deployment.apps/test-tomcat-deploy edited
[root@k8s-master test]#
```

该就绪探针会在容器中定期执行ls /var/ready命令。如果文件存在，ls命令会返回退出码0，否则返回一个非0退出码。

由于在执行kubectl edit之前，我的演示集群中已经有三个pod在运行了，并且修改Deployment的pod模板不会影响现有的pod，所以在添加就绪探针之后，正常情况下，Kubernetes会更新该Deployment对象，根据指定的replicas=3创建三个新的pod副本并终止现有的三个pod，但是在创建第一个新的pod后，由于该pod中就绪探针执行ls /var/ready命令返回的不是0（找不到这个文件），所以pod的READY状态一直显示0/1，因此既不会再创建第二个和第三个pod，也不会终止现有的pod，结果就会显示如下的情况：

```
[root@k8s-master test]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-6b8b5545b6-j45g5   1/1     Running   0          38h
test-tomcat-deploy-6b8b5545b6-rg9vc   1/1     Running   1          38h
test-tomcat-deploy-6b8b5545b6-zdddb   1/1     Running   1          38h
test-tomcat-deploy-7898789b6f-lkrp6   0/1     Running   0          6m41s
```

查看这个新建的、还未READY的pod：

kubectl describe pod test-tomcat-deploy-7898789b6f-lkrp6

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  3m38s                default-scheduler  Successfully assigned default/test-tomcat-deploy-7898789b6f-lkrp6 to k8s-node1
  Normal   Pulled     2d2h                 kubelet, k8s-node1  Container image "tomcat:8.5.34-jre8-alpine" already present on machine
  Normal   Created    2d2h                 kubelet, k8s-node1  Created container tomcat
  Normal   Started    2d2h                 kubelet, k8s-node1  Started container tomcat
  Warning  Unhealthy  2d2h (x18 over 2d2h)  kubelet, k8s-node1  Readiness probe failed: ls: /var/ready: No such file or directory
```

可以看到事件中记录了就绪探针的执行情况。

如果我们进入这个容器，手动创建/var/ready文件，就会发现这个pod会变成READY状态：

kubectl exec -it test-tomcat-deploy-7898789b6f-lkrp6 bash

或者执行执行kubectl exec -it test-tomcat-deploy-7898789b6f-lkrp6 -- touch /var/ready

```
[root@k8s-master test]# kubectl exec -it test-tomcat-deploy-7898789b6f-lkrp6 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
bash-4.4# ls /var/ready
ls: /var/ready: No such file or directory
bash-4.4# touch /var/ready
bash-4.4# ls /var/ready
/var/ready
bash-4.4# exit
exit
```

```
[root@k8s-master test]# kubectl get pod
NAME                                   READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-6b8b5545b6-j45g5    1/1     Running   0          38h
test-tomcat-deploy-6b8b5545b6-rg9vc    1/1     Running   1          38h
test-tomcat-deploy-6b8b5545b6-zdddb    1/1     Running   1          38h
test-tomcat-deploy-7898789b6f-lkrp6    0/1     Running   0          7m53s
[root@k8s-master test]# kubectl get pod
NAME                                   READY   STATUS              RESTARTS   AGE
test-tomcat-deploy-6b8b5545b6-j45g5    1/1     Running             0          38h
test-tomcat-deploy-6b8b5545b6-rg9vc    1/1     Terminating         1          38h
test-tomcat-deploy-6b8b5545b6-zdddb    1/1     Running             1          38h
test-tomcat-deploy-7898789b6f-2nwct    0/1     ContainerCreating   0          3s
test-tomcat-deploy-7898789b6f-lkrp6    1/1     Running             0          7m56s
[root@k8s-master test]# kubectl get pod
NAME                                   READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-6b8b5545b6-j45g5    1/1     Running   0          38h
test-tomcat-deploy-6b8b5545b6-zdddb    1/1     Running   1          38h
test-tomcat-deploy-7898789b6f-2nwct    0/1     Running   0          18s
test-tomcat-deploy-7898789b6f-lkrp6    1/1     Running   0          8m11s
```

从上面我们还可以发现，旧有的名为test-tomcat-deploy-6b8b5545b6-rg9vc的pod正在被终止掉，并且一个新的名为test-tomcat-deploy-7898789b6f-2nwct的pod正在被创建。

但是这个正在被创建的pod在创建后仍然会同第一个创建的pod一样显示0/1，因为pod默认都没有/var/ready文件，因此就绪探针认为pod还未准备就绪：

```
[root@k8s-master test]# kubectl get pod
NAME                                   READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-6b8b5545b6-j45g5    1/1     Running   0          39h
test-tomcat-deploy-6b8b5545b6-zdddb    1/1     Running   1          39h
test-tomcat-deploy-7898789b6f-2nwct    0/1     Running   0          17m
test-tomcat-deploy-7898789b6f-lkrp6    1/1     Running   0          24m
[root@k8s-master test]#
```

通过相同的方式，进入新建的pod中，手动创建/var/ready文件，则后续创建的pod都会变成ready状态，旧有的pod全部都会被终止并替换掉：

```
[root@k8s-master test]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-7898789b6f-2nwct   1/1     Running   0          23m
test-tomcat-deploy-7898789b6f-khn9g   1/1     Running   0          2m17s
test-tomcat-deploy-7898789b6f-lkrp6   1/1     Running   0          31m
[root@k8s-master test]#
```

如果我们此时将某个pod中的/var/ready文件删除掉，则过一会儿后（默认10s）该pod的READY列会变成0/1：

kubectl exec test-tomcat-deploy-7898789b6f-lkrp6 -- rm -f /var/ready

```
[root@k8s-master test]# kubectl get pod
NAME                                READY   STATUS    RESTARTS   AGE
test-tomcat-deploy-7898789b6f-2nwct   1/1     Running   0          43m
test-tomcat-deploy-7898789b6f-khn9g   1/1     Running   0          22m
test-tomcat-deploy-7898789b6f-lkrp6   0/1     Running   0          51m
```

在上面的例子中，之所以选择这样一个奇怪的就绪探针，是因为这样可以方便地创建和删除文件，从而灵活地控制就绪探针的结果。但是这个例子也只是为了演示就绪探针的功能。在项目实践中，就绪探针应该依据于应用程序能否收到客户端的请求来返回成功或者失败。

如果希望将pod从服务中移除，可以通过删除pod或者更改pod标签的方式执行，而不是通过手动地更改探针来实现。例如，某个pod有一个enabled=True标签，有一个服务的标签选择器中也包含这个标签，如果希望将这个pod从该服务中移除，可以通过将pod中的enabled=True标签移除来实现。

需要注意的两点：

- 如果没有给pod添加就绪探针，那么pod就会立即添加到服务的endpoints列表中。如果pod中的应用需要等待一定的时间才能开始接收外部请求的话，当客户端请求到达该服务时，这些请求就有可能转发到该pod上，虽然这个pod中的应用正在启动中而且没有准备好接收外部连接。客户端因此可能会收到“Connection refused”错误。因此我们最好总是指定一个就绪探针。
- 当pod被停止的时候，运行在其中的应用程序通常会在收到终止信号后立即停止接收连接。可能你会认为只要启动了关机程序，就需要使就绪探针返回失败，以便确保该pod从所有相关服务中移除，但这并不是必需的，因为一旦我们删除了这个pod，Kubernetes就会从所有服务中移除这个pod。