阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

关注“阿里巴巴云原生”公众号
获取第一手技术资料

第 11 讲

可观测性：你的应用健康吗？

莫源　阿里巴巴技术专家

1 需求来源

2 Liveness与Readness

3 问题诊断

4 应用远程调试

5 课后总结与实践

需求来源

当我们迁移应用到Kubernetes后，要如何保障应用健康稳定？

提高应用的可观测性，提高应用的可恢复能力

应用的状态可以从被实时观测
- 应用健康状态
- 应用资源使用
- 应用实时日志

应用出现问题需要降低影响范围，进行问题调试、诊断

应用出现问题可以通过自愈机制恢复

1 需求来源

2 Liveness与Readness

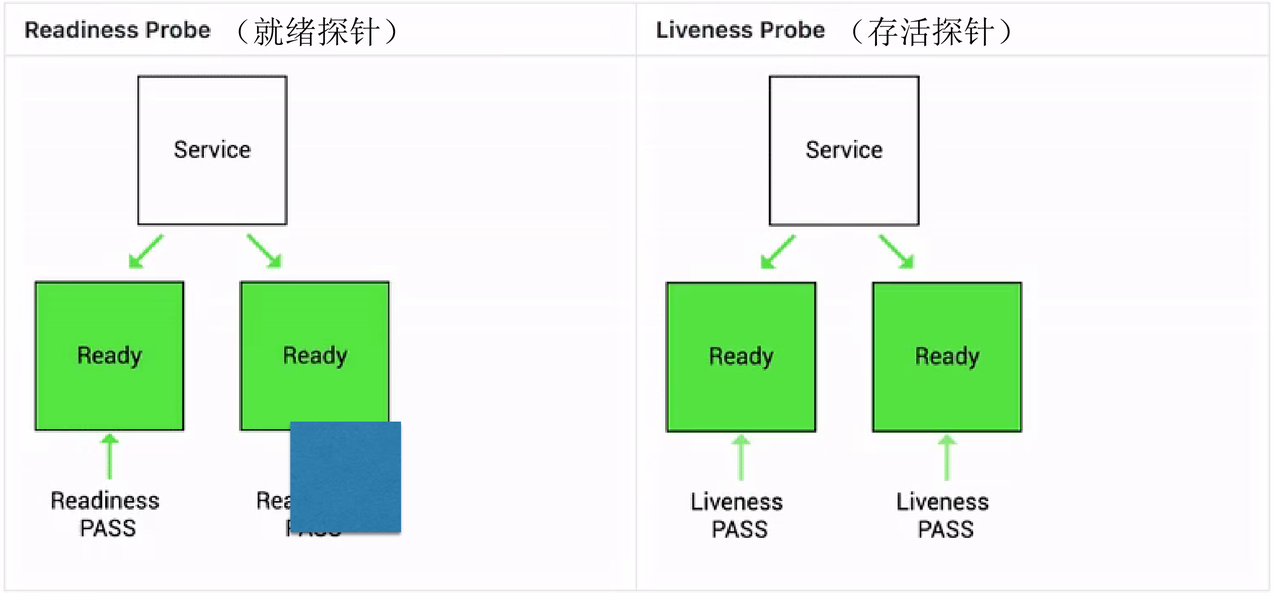
3 问题诊断

4 应用远程调试

5 课后总结与实践

应用健康状态 - 初识Liveness与Readness

当应用的Pod已经运行起来，如何让Kubernetes检查应用的状态，判断是否已经准备好对外提供服务？



应用健康状态 - 使用方式

探测方式

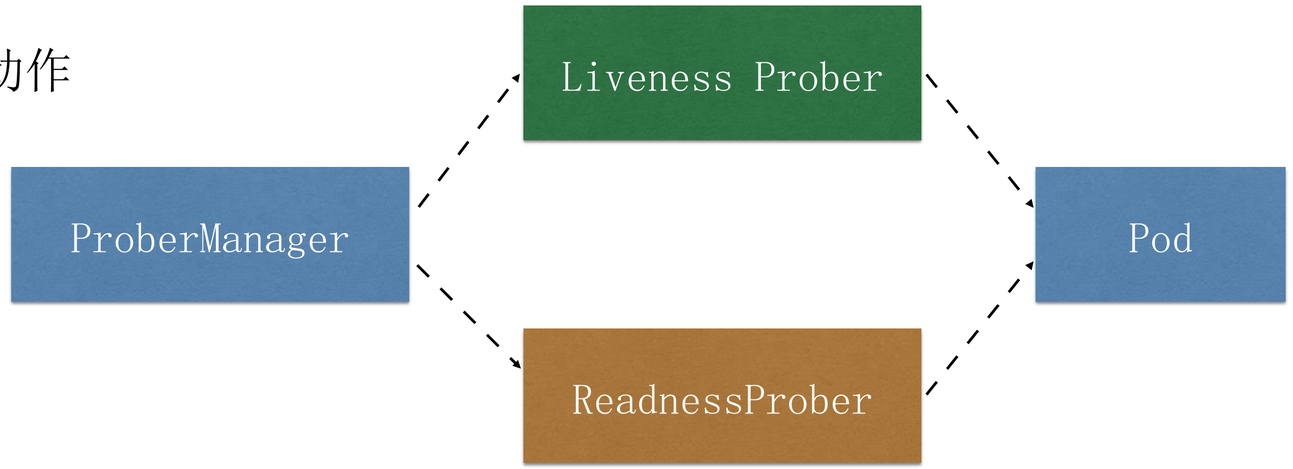
- httpGet　　通过发送http GET请求返回200-399状态码则表明容器健康。
- Exec　　通过执行命令来检查服务是否正常，命令返回值为0则表示容器健康。
- tcpSocket　　通过容器的IP和Port执行TCP检查，如果能够建立TCP连接则表明容器健康。

探测结果

- Success　　Container通过了检查
- Failure　　Container未通过检查
- Unknown　　未能执行检查，不采取任何动作

重启策略

- Always　　总是重启
- OnFailure　　失败才重启
- Never　　永远不重启



应用健康状态 - Pod Probe Spec

exec

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/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
```

httpGet

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
        httpHeaders:
        - name: Custom-Header
          value: Awesome
      initialDelaySeconds: 3
      periodSeconds: 3
```

tcpSocket

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: goproxy
  labels:
    app: goproxy
spec:
  containers:
  - name: goproxy
    image: k8s.gcr.io/goproxy:0.1
    ports:
    - containerPort: 8080
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
      periodSeconds: 10
    livenessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 20
```

其他参数

- initialDelaySeconds　　Pod启动后延迟多久进行检查，单位：秒
- periodSeconds　　检查的间隔时间，默认为10秒，单位：秒
- timeoutSeconds　　探测的超时时间，默认为1秒，单位：秒
- successThreshold　　探测失败后再次判断成功的阈值，默认为1，单位：次
- failureThreshold　　探测失败的重试次数，默认为3，单位：次

应用健康状态 - Liveness与Readness的总结

	Liveness （存活探针）	Readness（就绪探针）
介绍	用于判断容器是否存活，即Pod状态是否为Running，如果Liveness探针判断容器不健康，则会触发kubelet杀掉容器，并根据配置的策略判断是否重启容器，如果默认不配置Liveness探针，则认为返回值默认为成功。	用于判断容器是否启动完成，即Pod的Condition是否为Ready，如果探测结果不成功，则会将Pod从Endpoint中移除，直至下次判断成功，再将Pod挂回到Endpoint上。
检测失败	杀掉Pod	切断上层流量到Pod
适用场景	支持重新拉起的应用	启动后无法立即对外服务的应用
注意事项	不论是Liveness还是Readness探针，选择合适的探测方式可以防止被误操作： 1. 调大判断的超时阈值，防止在容器压力较高的情况下出现偶发超时 2. 调整判断的次数阈值，3次的默认值在短周期下不一定是最佳实践 3. exec的如果执行的是shell脚本判断，在容器中可能调用时间会非常长 4. 使用tcpSocket的方式遇到TLS的场景，需要业务层判断是否有影响	

1 需求来源

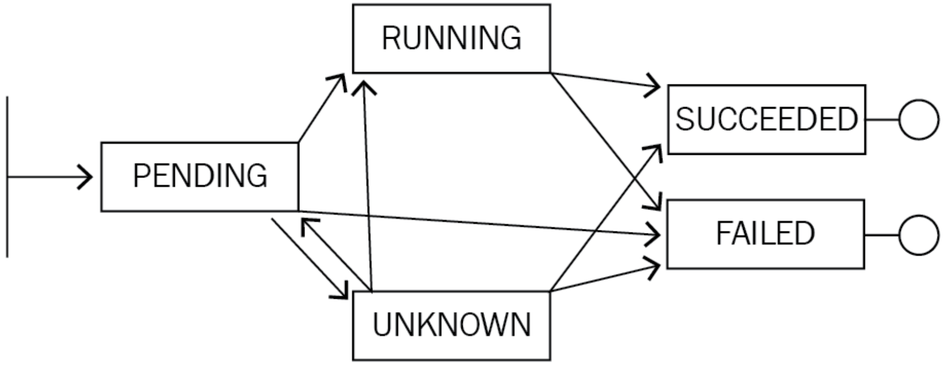
2 Liveness与Readiness

3 问题诊断

4 应用远程调试

5 课后总结与实践

应用故障排查 - 了解状态机制



Value	Description
`Pending`	The Pod has been accepted by the Kubernetes system, but one or more of the Container images has not been created. This includes time before being scheduled as well as time spent downloading images over the network, which could take a while.
`Running`	The Pod has been bound to a node, and all of the Containers have been created. At least one Container is still running, or is in the process of starting or restarting.
`Succeeded`	All Containers in the Pod have terminated in success, and will not be restarted.
`Failed`	All Containers in the Pod have terminated, and at least one Container has terminated in failure. That is, the Container either exited with non-zero status or was terminated by the system.
`Unknown`	For some reason the state of the Pod could not be obtained, typically due to an error in communicating with the host of the Pod.

```
Name:               nginx-deployment-basic-c4598964d-tjjnn
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               cn-beijing.192.168.3.167/192.168.3.167
Start Time:         Tue, 02 Jul 2019 10:00:40 +0800
Labels:             app=nginx
                    pod-template-hash=c4598964d
Annotations:        <none>
Status:             Pending
IP:                 172.20.1.214
Controlled By:      ReplicaSet/nginx-deployment-basic-c4598964d
Containers:
  nginx:
    Container ID:
    Image:          nginx:ne
    Image ID:
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Waiting
      Reason:       ImagePullBackOff
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-nnmg8 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-nnmg8:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-nnmg8
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type     Reason     Age                From                                Message
  ----     ------     ----               ----                                -------
  Normal   Scheduled  37s                default-scheduler                   Successfully assigned default/nginx-deployment-basic-c4598964d-tjjnn to cn-beijing.192.168.3.167
  Normal   BackOff    31s                kubelet, cn-beijing.192.168.3.167   Back-off pulling image "nginx:ne"
  Warning  Failed     31s                kubelet, cn-beijing.192.168.3.167   Error: ImagePullBackOff
  Normal   Pulling    16s (x2 over 36s)  kubelet, cn-beijing.192.168.3.167   pulling image "nginx:ne"
  Warning  Failed     10s (x2 over 31s)  kubelet, cn-beijing.192.168.3.167   Failed to pull image "nginx:ne": rpc error: code = Unknown desc = Error response from daemon: manifest f
  Warning  Failed     10s (x2 over 31s)  kubelet, cn-beijing.192.168.3.167   Error: ErrImagePull
```

应用故障排查 - 常见应用异常

Pod停留在Pending

Pending表示调度器没有介入，可以通过kubectl describe pod，查看事件排查，通常和资源使用相关。

Pod停留在waiting

一般表示Pod的镜像没有正常的拉取，通常可能和私有镜像拉取，镜像地址不存在，镜像公网拉取相关。

Pod不断被拉起且可以看到crashing

通常表示Pod已经完成调度并启动，但是启动失败，通常是由于配置、权限造成，需查看Pod日志。

Pod处在Running但是没有正常工作

通常是由于部分字段拼写错误造成的，可以通过校验部署来排查，例如：kubectl apply —validate -f pod.yaml

Service无法正常工作

在排除网络插件自身的问题外，最可能的是label配置有问题，可以通过查看endpoint的方式进行检查。

1 需求来源

2 Liveness与Readness

3 问题诊断

4 应用远程调试

5 课后总结与实践

应用远程调试 - Pod远程调试

当我们把一个应用部署到集群中发现问题需要快速进行验证修改时，可以通过远程调试的方式进行修改、验证。

进入一个正在运行的Pod

```
kubectl exec -it pod-name /bin/bash
```

进入一个正在运行包含多容器的Pod

```
kubectl exec -it pod-name -c container-name /bin/bash
```

应用远程调试 - Service远程调试

当集群中应用依赖的应用需要本地调试时：
可以使用Telepresence将本地的应用代理到集群中的一个Service上。

```
Telepresence  —swap-deployment $DEPLOYMENT_NAME
```

当本地开发的应用需要调用集群中的服务时：
可以使用Port-Forward将远程的应用代理到本地的端口上。

```
kubectl port-forward svc/app -n app-namespace
```

开源的调试工具 - kubectl-debug

1 需求来源

2 Liveness与Readness

3 问题诊断

4 应用远程调试

5 课后总结与实践

课后总结与实践

应用状态探针与自动恢复

1. Liveness Probe保活探针
2. Readness Probe就绪探针

应用问题诊断的三个步骤

1. 通过describe查看状态，通过状态判断排查方向。
2. 查看对象的event事件，获取更详细的信息。
3. 查看Pod的日志确定应用自身的情况。

应用远程调试

1. 代理本地应用到集群 - Telepresence
2. 代理远程应用到本地 - Port-Forward

关注“阿里巴巴云原生”公众号
获取第一手技术资料