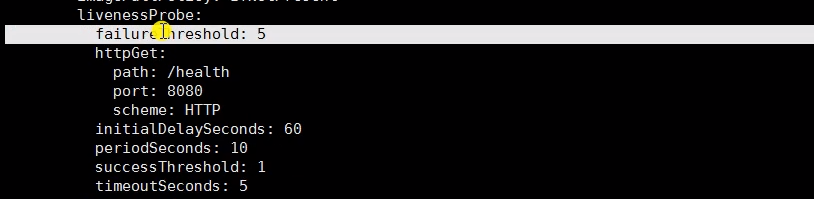
## 视频更正

注意：视频中讲解的容器重启间隔计算方法有误（零宕机必备知识：StartupProbe小结，第07:00分钟处），如下：



课程讲解的重启时间是10\*5\*5（periodSeconds \* failureThreshold \* timeoutSeconds）

# initialDelaySeconds: 60 # 初始化时间

# timeoutSeconds: 5 # 超时时间

# periodSeconds: 10 # 检测间隔

# successThreshold: 1 # 检查成功为1次表示就绪

# failureThreshold: 5 # 检测失败2次表示未就绪

此处讲解是错误的，正确的时间：每次检查的间隔是10秒，最长超时时间是5秒，也就是单次检查应该是10 + 5 = 15秒（periodSeconds + timeoutSeconds），并不是10 \* 5

所以最长的重启时间为（10 + 5）\* 5

（periodSeconds + timeoutSeconds） \* failureThreshold

此时又分为了两种情况：

1. **首次启动时**：最长重启时间需要加上initialDelaySeconds，因为需要等待initialDelaySeconds秒后才会执行健康检查。最长重启时间：（periodSeconds + timeoutSeconds） \* failureThreshold + initialDelaySeconds
2. **程序启动完成后**：此时不需要计入initialDelaySeconds，最长重启时间：（periodSeconds + timeoutSeconds） \* failureThreshold

## 什么是Pod？

Pod是Kubernetes中最小的单元，它由一组、一个或多个容器组成，每个Pod还包含了一个Pause容器，Pause容器是Pod的父容器，主要负责僵尸进程的回收管理，通过Pause容器可以使同一个Pod里面的多个容器共享存储、网络、PID、IPC等。

## 定义一个Pod

apiVersion: v1 # 必选，API的版本号

kind: Pod # 必选，类型Pod

metadata: # 必选，元数据

name: nginx # 必选，符合RFC 1035规范的Pod名称

namespace: default # 可选，Pod所在的命名空间，不指定默认为default，可以使用-n 指定namespace

labels: # 可选，标签选择器，一般用于过滤和区分Pod

app: nginx

role: frontend # 可以写多个

annotations: # 可选，注释列表，可以写多个

app: nginx

spec: # 必选，用于定义容器的详细信息

initContainers: # 初始化容器，在容器启动之前执行的一些初始化操作

- command:

- sh

- -c

- echo "I am InitContainer for init some configuration"

image: busybox

imagePullPolicy: IfNotPresent

name: init-container

containers: # 必选，容器列表

- name: nginx # 必选，符合RFC 1035规范的容器名称

image: nginx:latest # 必选，容器所用的镜像的地址

imagePullPolicy: Always # 可选，镜像拉取策略

command: # 可选，容器启动执行的命令

- nginx

- -g

- "daemon off;"

workingDir: /usr/share/nginx/html # 可选，容器的工作目录

volumeMounts: # 可选，存储卷配置，可以配置多个

- name: webroot # 存储卷名称

mountPath: /usr/share/nginx/html # 挂载目录

readOnly: true # 只读

ports: # 可选，容器需要暴露的端口号列表

- name: http # 端口名称

containerPort: 80 # 端口号

protocol: TCP # 端口协议，默认TCP

env: # 可选，环境变量配置列表

- name: TZ # 变量名

value: Asia/Shanghai # 变量的值

- name: LANG

value: en\_US.utf8

resources: # 可选，资源限制和资源请求限制

limits: # 最大限制设置

cpu: 1000m

memory: 1024Mi

requests: # 启动所需的资源

cpu: 100m

memory: 512Mi

# startupProbe: # 可选，检测容器内进程是否完成启动。注意三种检查方式同时只能使用一种。

# httpGet: # httpGet检测方式，生产环境建议使用httpGet实现接口级健康检查，健康检查由应用程序提供。

# path: /api/successStart # 检查路径

# port: 80

readinessProbe: # 可选，健康检查。注意三种检查方式同时只能使用一种。

httpGet: # httpGet检测方式，生产环境建议使用httpGet实现接口级健康检查，健康检查由应用程序提供。

path: / # 检查路径

port: 80 # 监控端口

livenessProbe: # 可选，健康检查

#exec: # 执行容器命令检测方式

#command:

#- cat

#- /health

#httpGet: # httpGet检测方式

# path: /\_health # 检查路径

# port: 8080

# httpHeaders: # 检查的请求头

# - name: end-user

# value: Jason

tcpSocket: # 端口检测方式

port: 80

initialDelaySeconds: 60 # 初始化时间

timeoutSeconds: 2 # 超时时间

periodSeconds: 5 # 检测间隔

successThreshold: 1 # 检查成功为2次表示就绪

failureThreshold: 2 # 检测失败1次表示未就绪

lifecycle:

postStart: # 容器创建完成后执行的指令, 可以是exec httpGet TCPSocket

exec:

command:

- sh

- -c

- 'mkdir /data/ '

preStop:

httpGet:

path: /

port: 80

# exec:

# command:

# - sh

# - -c

# - sleep 9

restartPolicy: Always # 可选，默认为Always

#nodeSelector: # 可选，指定Node节点

# region: subnet7

imagePullSecrets: # 可选，拉取镜像使用的secret，可以配置多个

- name: default-dockercfg-86258

hostNetwork: false # 可选，是否为主机模式，如是，会占用主机端口

volumes: # 共享存储卷列表

- name: webroot # 名称，与上述对应

emptyDir: {} # 挂载目录

#hostPath: # 挂载本机目录

# path: /etc/hosts

apiVersion: v1 # 必选，API的版本号

kind: Pod # 必选，类型Pod

metadata: # 必选，元数据

name: nginx # 必选，符合RFC 1035规范的Pod名称

# namespace: default # 可选，Pod所在的命名空间，不指定默认为default，可以使用-n 指定namespace

labels: # 可选，标签选择器，一般用于过滤和区分Pod

app: nginx

role: frontend # 可以写多个

annotations: # 可选，注释列表，可以写多个

app: nginx

spec: # 必选，用于定义容器的详细信息

# initContainers: # 初始化容器，在容器启动之前执行的一些初始化操作

# - command:

# - sh

# - -c

# - echo "I am InitContainer for init some configuration"

# image: busybox

# imagePullPolicy: IfNotPresent

# name: init-container

containers: # 必选，容器列表

- name: nginx # 必选，符合RFC 1035规范的容器名称

image: nginx:1.15.2 # 必选，容器所用的镜像的地址

imagePullPolicy: IfNotPresent # 可选，镜像拉取策略, IfNotPresent: 如果宿主机有这个镜像，那就不需要拉取了. Always: 总是拉取, Never: 不管是否存储都不拉去

command: # 可选，容器启动执行的命令 ENTRYPOINT, arg --> cmd

- nginx

- -g

- "daemon off;"

workingDir: /usr/share/nginx/html # 可选，容器的工作目录

# volumeMounts: # 可选，存储卷配置，可以配置多个

# - name: webroot # 存储卷名称

# mountPath: /usr/share/nginx/html # 挂载目录

# readOnly: true # 只读

ports: # 可选，容器需要暴露的端口号列表

- name: http # 端口名称

containerPort: 80 # 端口号

protocol: TCP # 端口协议，默认TCP

env: # 可选，环境变量配置列表

- name: TZ # 变量名

value: Asia/Shanghai # 变量的值

- name: LANG

value: en\_US.utf8

# resources: # 可选，资源限制和资源请求限制

# limits: # 最大限制设置

# cpu: 1000m

# memory: 1024Mi

# requests: # 启动所需的资源

# cpu: 100m

# memory: 512Mi

# startupProbe: # 可选，检测容器内进程是否完成启动。注意三种检查方式同时只能使用一种。

# httpGet: # httpGet检测方式，生产环境建议使用httpGet实现接口级健康检查，健康检查由应用程序提供。

# path: /api/successStart # 检查路径

# port: 80

# readinessProbe: # 可选，健康检查。注意三种检查方式同时只能使用一种。

# httpGet: # httpGet检测方式，生产环境建议使用httpGet实现接口级健康检查，健康检查由应用程序提供。

# path: / # 检查路径

# port: 80 # 监控端口

# livenessProbe: # 可选，健康检查

#exec: # 执行容器命令检测方式

#command:

#- cat

#- /health

#httpGet: # httpGet检测方式

# path: /\_health # 检查路径

# port: 8080

# httpHeaders: # 检查的请求头

# - name: end-user

# value: Jason

# tcpSocket: # 端口检测方式

# port: 80

# initialDelaySeconds: 60 # 初始化时间

# timeoutSeconds: 2 # 超时时间

# periodSeconds: 5 # 检测间隔

# successThreshold: 1 # 检查成功为2次表示就绪

# failureThreshold: 2 # 检测失败1次表示未就绪

# lifecycle:

# postStart: # 容器创建完成后执行的指令, 可以是exec httpGet TCPSocket

# exec:

# command:

# - sh

# - -c

# - 'mkdir /data/ '

# preStop:

# httpGet:

# path: /

# port: 80

# exec:

# command:

# - sh

# - -c

# - sleep 9

restartPolicy: Always # 可选，默认为Always，容器故障或者没有启动成功，那就自动该容器，Onfailure: 容器以不为0的状态终止，自动重启该容器, Never:无论何种状态，都不会重启

#nodeSelector: # 可选，指定Node节点

# region: subnet7

# imagePullSecrets: # 可选，拉取镜像使用的secret，可以配置多个

# - name: default-dockercfg-86258

# hostNetwork: false # 可选，是否为主机模式，如是，会占用主机端口

# volumes: # 共享存储卷列表

# - name: webroot # 名称，与上述对应

# emptyDir: {} # 挂载目录

# #hostPath: # 挂载本机目录

# # path: /etc/hosts

#

## Pod探针

* StartupProbe：k8s1.16版本后新加的探测方式，用于判断容器内应用程序是否已经启动。如果配置了startupProbe，就会先禁止其他的探测，直到它成功为止，成功后将不在进行探测。
* LivenessProbe：用于探测容器是否运行，如果探测失败，kubelet会根据配置的重启策略进行相应的处理。若没有配置该探针，默认就是success。
* ReadinessProbe：一般用于探测容器内的程序是否健康，它的返回值如果为success，那么久代表这个容器已经完成启动，并且程序已经是可以接受流量的状态。

## Pod探针的检测方式

* ExecAction：在容器内执行一个命令，如果返回值为0，则认为容器健康。
* TCPSocketAction：通过TCP连接检查容器内的端口是否是通的，如果是通的就认为容器健康。
* HTTPGetAction：通过应用程序暴露的API地址来检查程序是否是正常的，如果状态码为200~400之间，则认为容器健康。

## 探针检查参数配置

# initialDelaySeconds: 60 # 初始化时间

# timeoutSeconds: 2 # 超时时间

# periodSeconds: 5 # 检测间隔

# successThreshold: 1 # 检查成功为1次表示就绪

# failureThreshold: 2 # 检测失败2次表示未就绪

Prestop：先去请求eureka接口，把自己的IP地址和端口号，进行下线，eureka从注册表中删除该应用的IP地址。然后容器进行sleep 90；kill `pgrep java`