

K8S Pod Yaml

YAML特点

YAML 的英文全称是："Yet Another Markup Language"

YAML有下面特点：

- 层次分明、结构清晰
- 使用简单、上手容易
- 表达强大、语义丰富

但是要注意几点：

- 大小写敏感
- 禁止使用tab键缩进，只能空格键

Pod中的yaml参数

基本配置

```

apiVersion: v1                                #必选，版本号，例如v1，可以用 kubectl api-versions
查询到
kind: Pod                                     #必选，指yaml文件定义的k8s 资源类型或角色，比如：Pod
metadata:                                     #必选，元数据对象
  name: string                               #必选，元数据对象的名字，自己定义，比如命名Pod的名字
  namespace: string                          #必选，元数据对象的名称空间，默认为"default"
  labels:                                    #自定义标签
    key1: value1                             #自定义标签键值对1
    key2: value2                             #自定义标签键值对2
spec:                                         #必选，对象【如pod】的详细定义
  containers:                               #必选，spec对象的容器信息
  - name: string                             #必选，容器名称
    image: string                            #必选，要用到的镜像名称
    ports:                                   #需要暴露的端口号列表
    - name: string                           #端口的名称
      containerPort: int                    #容器监听的端口号
      #除非绝对必要，否则不要为 Pod 指定 hostPort。 将 Pod 绑定到hostPort时，它会限制
Pod 可以调度的位置数
      #DaemonSet 中的 Pod 可以使用 hostPort，从而可以通过节点 IP 访问到 Pod；因为
DaemonSet模式下Pod不会被调度到其他节点。
      #一般情况下 containerPort与hostPort值相同
      hostPort: int                         #可以通过宿主机+hostPort的方式访问该Pod。例如：pod在/调度到了
k8s-node02 【172.16.1.112】， hostPort为8090，那么该Pod可以通过172.16.1.112:8090方式进行访问。

```

实例

Pod类型:

- 单容器pod
- 多容器pod

创建单容器pod

可以使用kubctl run命令简单地创建

```
$ kubect1 run <name of pod> --image=<name of the image from docker>

# 示例-我们将创建一个带有tomcat映像的容器
$ kubect1 run tomcat --image = tomcat:8.0
```

也可以通过创建yaml文件，然后运行kubect1 create命令来完成此操作。

```
apiVersion: v1
kind: Pod
metadata:
  name: Tomcat
spec:
  containers:
  - name: Tomcat
    image: tomcat: 8.0
    ports:
      containerPort: 8080
    imagePullPolicy: Always

# 创建上述yaml文件后，我们将使用tomcat.yml名称保存该文件，然后运行create命令运行该文档。
$ kubect1 create -f tomcat.yml

# 它将创建一个名为tomcat的pod。我们可以将describe命令与kubect1一起使用来描述pod。
```

创建多容器pod

使用带有容器定义的yaml邮件创建多容器容器。

```
apiVersion: v1
kind: Pod
metadata:
  name: Tomcat
spec:
  containers:
  - name: Tomcat
    image: tomcat: 8.0
    ports:
      containerPort: 8081
```

```

    imagePullPolicy: Always
  - name: Database
    image: mongoDB
    ports:
      containerPort: 8091
    imagePullPolicy: Always

```

在上面的代码中，我们创建了一个pod，其中包含两个容器，一个容器用于tomcat，另一个容器用于MongoDB。

Pod yaml配置详解

```

apiVersion: v1                                #必选，版本号，例如v1，可以用 kubectl api-versions
查询到
kind: Pod                                     #必选，指yaml文件定义的k8s 资源类型或角色，比如：Pod
metadata:                                     #必选，元数据对象
  name: string                               #必选，元数据对象的名字，自己定义，比如命名Pod的名字
  namespace: string                           #必选，元数据对象的名称空间，默认为"default"
  labels:                                     #自定义标签
    key1: value1                             #自定义标签键值对1
    key2: value2                             #自定义标签键值对2
  annotations:                              #自定义注解
    key1: value1                             #自定义注解键值对1
    key2: value2                             #自定义注解键值对2
spec:                                         #必选，对象【如pod】的详细定义
  containers:                               #必选，spec对象的容器信息
    - name: string                           #必选，容器名称
      image: string                          #必选，要用到的镜像名称
      imagePullPolicy: [Always|Never|IfNotPresent] #获取镜像的策略；(1)Always：意思是
每次都尝试重新拉取镜像；(2)Never：表示仅使用本地镜像，即使本地没有镜像也不拉取；(3)
IfNotPresent：如果本地有镜像就使用本地镜像，没有就拉取远程镜像。默认：Always
      command: [string]                     #指定容器启动命令，由于是数组因此可以指定多个。不指定则
使用镜像打包时指定的启动命令。
      args: [string]                         #指定容器启动命令参数，由于是数组因此可以指定多个
      workingDir: string                     #指定容器的工作目录
      volumeMounts:                          #指定容器内部的存储卷配置
        - name: string                       #指定可以被容器挂载的存储卷的名称。跟下面volume字段的
name值相同表示使用这个存储卷
        mountPath: string                    #指定可以被容器挂载的存储卷的路径，应少于512字符
        readOnly: boolean                   #设置存储卷路径的读写模式，true或者false，默认为读写模
式false
      ports:                                #需要暴露的端口号列表
        - name: string                       #端口的名称
          containerPort: int                 #容器监听的端口号
          #除非绝对必要，否则不要为 Pod 指定 hostPort。 将 Pod 绑定到hostPort时，它会限制
Pod 可以调度的位置数
          #DaemonSet 中的 Pod 可以使用 hostPort，从而可以通过节点 IP 访问到 Pod；因为
DaemonSet模式下Pod不会被调度到其他节点。
          #一般情况下 containerPort与hostPort值相同
          hostPort: int                      #可以通过宿主机+hostPort的方式访问该Pod。例如：pod在/调度到了
k8s-node02 【172.16.1.112】， hostPort为8090，那么该Pod可以通过172.16.1.112:8090方式进

```

行访问。

```

    protocol: string          #端口协议，支持TCP和UDP，默认TCP
    env:                      #容器运行前需设置的环境变量列表
    - name: string            #环境变量名称
      value: string           #环境变量的值
    resources:                #资源限制和资源请求的设置（设置容器的资源上限）
      limits:                 #容器运行时资源使用的上限
        cpu: string           #CPU限制，单位为core数，允许浮点数，如0.1等价于100m，
                                0.5等价于500m；因此如果小于1那么优先选择如100m的形式，精度为1m。这个数字用作 docker run
                                命令中的 --cpu-quota 参数。
        memory: string        #内存限制，单位：E,P,T,G,M,K；或者
                                Ei,Pi,Ti,Gi,Mi,Ki；或者字节数。将用于docker run --memory参数
      requests:               #容器启动和调度时的限制设定
        cpu: string           #CPU请求，容器启动时初始化可用数量，单位为core数，允许
                                浮点数，如0.1等价于100m，0.5等价于500m；因此如果小于1那么优先选择如100m的形式，精度为
                                1m。这个数字用作 docker run 命令中的 --cpu-shares 参数。
        memory: string        #内存请求，容器启动的初始化可用数量。单位：
                                E,P,T,G,M,K；或者Ei,Pi,Ti,Gi,Mi,Ki；或者字节数
    # 参见官网地址：https://kubernetes.io/zh/docs/tasks/configure-pod-
    container/configure-liveness-readiness-startup-probes/
    livenessProbe:            #对Pod内各容器健康检查的设置，当探测无响应几次后将自动
                                重启该容器，检查方法有exec、httpGet和tcpSocket，对一个容器【只需设置其中一种方法即可】
    exec:                     #对Pod内容器健康检查方式设置为exec方式
      command: [string]       #exec方式需要制定的命令或脚本
    httpGet:                  #对Pod内容器健康检查方法设置为HttpGet，需要制定Path、
    port                       #访问 HTTP 服务的路径
      path: string             #访问容器的端口号或者端口名。如果数字必须在 1 ~ 65535
      port: number             之间。
    host: string               #当没有定义 "host" 时，使用 "PodIP"
    scheme: string             #当没有定义 "scheme" 时，使用 "HTTP"，scheme 只允许
    "HTTP" 和 "HTTPS"
    httpHeaders:               #请求中自定义的 HTTP 头。HTTP 头字段允许重复。
      - name: string
        value: string
    tcpSocket:                 #对Pod内容器健康检查方式设置为tcpSocket方式
      port: number
    initialDelaySeconds: 5     #容器启动完成后，kubelet在执行第一次探测前应该等待 5
    秒。默认是 0 秒，最小值是 0。
    periodSeconds: 60          #指定 kubelet 每隔 60 秒执行一次存活探测。默认是 10
    秒。最小值是 1
    timeoutSeconds: 3           #对容器健康检查探测等待响应的超时时间为 3 秒，默认1秒
    successThreshold: 1         #检测到有1次成功则认为服务是`就绪`
    failureThreshold: 5         #检测到有5次失败则认为服务是`未就绪`。默认值是 3，最小
    值是 1。
    restartPolicy: [Always|Never|OnFailure] #Pod的重启策略，默认Always。Always表示一
    旦不管以何种方式终止运行，kubelet都将重启；OnFailure表示只有Pod以非0退出码退出才重启；
    Nerver表示不再重启该Pod
    nodeSelector:              #定义Node的label过滤标签，以key: value的格式指定。节
    点选择，先给主机打标签kubect1 label nodes kube-node01 key1=value1
      key1: value1
    imagePullSecrets:          #Pull镜像时使用的secret名称，以name: secretKeyName格
    式指定
      - name: string

```

```

    hostNetwork: false          #是否使用主机网络模式，默认为false。如果设置为true，表示使用宿主机网络，不使用docker网桥
# volumes 和 containers 是同层级 *****
# 参见官网地址: https://kubernetes.io/zh/docs/concepts/storage/volumes/
volumes:                       #定义了paues容器关联的宿主机或分布式文件系统存储卷列表
                                (volumes类型有很多种，选其中一种即可)
  - name: string                #共享存储卷名称。
    emptyDir: {}                #类型为emptyDir的存储卷，与Pod同生命周期的一个临时目录。
                                当Pod因为某些原因被从节点上删除时，emptyDir卷中的数据也会永久删除。
    hostPath: string            #类型为hostPath的存储卷，表示挂载Pod所在宿主机的文件或目录
                                录
    path: string                #在宿主机上文件或目录的路径
    type: [ |DirectoryOrCreate|Directory|FileOrCreate|File] #空字符串（默认）用于向后兼容，这意味着在安装 hostPath 卷之前不会执行任何检查。DirectoryOrCreate: 如果给定目录不存在则创建，权限设置为 0755，具有与 Kubelet 相同的组和所有权。Directory: 给定目录必须存在。FileOrCreate: 如果给定文件不存在，则创建空文件，权限设置为 0644，具有与 Kubelet 相同的组和所有权。File: 给定文件必须存在。
  secret:                       #类型为secret的存储卷，挂载集群预定义的secre对象到容器内部。Secret 是一种包含少量敏感信息例如密码、token 或 key 的对象。放在一个 secret 对象中可以更好地控制它的用途，并降低意外暴露的风险。
    secretName: string           #secret 对象的名字
    items:                       #可选，修改key 的目标路径
      - key: username            #username secret存储在/etc/foo/my-group/my-username
                                文件中而不是 /etc/foo/username 中。【此时存在
                                spec.containers[].volumeMounts[].mountPath为/etc/foo】
    path: my-group/my-username
  configMap:                     #类型为configMap的存储卷，挂载预定义的configMap对象到容器内部。ConfigMap 允许您将配置文件与镜像文件分离，以使容器化的应用程序具有可移植性。
    name: string                 #提供你想要挂载的 ConfigMap 的名字

```