链滴

Kubernetes 核心原理 --- Pod

作者：etscript

原文链接：https://ld246.com/article/1563592413397

来源网站：链滴

本文主要参考了以下几篇文章，将一些基础的概念整理了一下

Kubernetes架构

Kubernetes指南

不完美的 K8S 与阿里的解决之道

上一篇对Kubernetes的基础架构整理了一下，这篇主要讲Pod，其他 Api Server、Controller Manaer、Scheduler、Kubelet 的原理这里不提及：

1. Pod 的介绍
2. Pod 的相关命令
3. Pod 的定义文件
4. Pod 的配置管理

1. Pod 的介绍:

1.1. Pod 的概念

- Pod是kubernetes集群中最小的部署和管理的 基本单元，协同寻址，协同调度。
- Pod是一个或多个容器的集合，是一个或一组服务（进程）的抽象集合。
- Pod中可以共享网络和存储（可以简单理解为一个逻辑上的虚拟机，但并不是虚拟机）。
- Pod被创建后用一个 UID来唯一标识，当Pod生命周期结束，被一个等价Pod替代，UID将重新生。

1.2. Pod 的概念

- Docker是目前Pod最常用的容器环境，但仍支持其他容器环境。
- Pod是一组被模块化的拥有共享命名空间和共享存储卷的容器，但并没有共享PID 命名空间（即同Pod的不同容器中进程的PID是独立的，互相看不到非自己容器的进程）。

1.3 . Pod中容器的运行方式

1. **只运行一个单独的容器**

即one-container-per-Pod模式，是最常用的模式，可以把这样的Pod看成单独的一个容器去管理。

2. **运行多个强关联的容器**

即sidecar模式，Pod 封装了一组紧耦合、共享资源、协同寻址的容器，将这组容器作为一个管理单。

2. Pod 的相关命令:

操作	命令
创建 yaml	kubectl create -f frontend-localredis-pod

原文链接：Kubernetes 核心原理 --- Pod

查询Pod运行状态 kubectl get pods -n <NAMESPACE>

查询Pod详情 kebectl describe pod <POD_NAME> -n <NAMESPACE>

删除 kubectl delete pod <POD_NAME> 、kubectl delete pod --all

更新 kubectl replace pod.yaml

3. Pod 的定义文件：

```
apiVersion: v1
kind: Pod
metadata:
  name: string
  namaspace: string
  labels:
  - name: string
      annotations:
  - name: string
  spec:
    containers:
  - name: string
    images: string
    imagePullPolice: [Always | Never | IfNotPresent]
    command: [string]
    args: [string]
    workingDir: string
    volumeMounts:
    - name: string
      mountPath: string
      readOnly: boolean
      ports:
    - name: string
      containerPort: int
      hostPort: int
      protocol: string
      env:
    - name: string
      value: string
      resources:
      limits:
        cpu: string
        memory: string
      requests:
        cpu: string
        memory: string
      livenessProbe:
      exec:
        command: [string]
      httpGet:
        path: string
        port: int
```

```
        host: string
        scheme: string
        httpHeaders:
        - name: string
          value: string
          tcpSocket:
            port: int
          initialDelaySeconds: number
          timeoutSeconds: number
          periodSeconds: number
          successThreshold: 0
          failureThreshold: 0
          securityContext:
          privileged: false
          restartPolicy: [Always | Never | OnFailure]
          nodeSelector: object
          imagePullSecrets:
- name: string
    hostNetwork: false
      volumes:
- name: string
  emptyDir: {}
  hostPath:
    path: string
  secret:
    secretName: string
    items:
    - key: string
      path: string
      configMap:
        name: string
        items:
    - key: string
      path: string
```

4. Pod 的配置管理

4.1. ConfigMap：容器应用的配置管理

使用场景：

1. 生成为容器内的环境变量。
2. 设置容器启动命令的启动参数（需设置为环境变量）。
3. 以Volume的形式挂载为容器内部的文件或目录。

4.2. 创建ConfigMap

cm-appvars.yaml

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: cm-appvars
data:
  apploglevel: info
  appdatadir: /var/data
```

常用命令

```
kubectl create -f cm-appvars.yaml

kubectl get configmap

kubectl describe configmap cm-appvars

kubectl get configmap cm-appvars -o yaml
```

可以通过yaml配置文件或者使用kubectl create configmap命令的方式创建ConfigMap。

4.3. ConfigMap和Pod关联

ConfigMap的yaml文件:cm-appvars.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm-appvars
data:
  apploglevel: info
  appdatadir: /var/data
```

Pod的yaml文件：cm-test-pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: cm-test-pod
spec:
  containers:
  - name: cm-test
    image: busybox
    command: ["/bin/sh","-c","env|grep APP"]
    env:
    - name: APPLOGLEVEL
      valueFrom:
        configMapKeyRef:
          name: cm-appvars
          key: apploglevel
    - name: APPDATADIR
      valueFrom:
        configMapKeyRef:
          name: cm-appvars
          key: appdatadir
```

创建命令：

kubectl create -f cm-test-pod.yaml

kubectl get pods --show-all

kubectl logs cm-test-pod

点到为止，不过度深入，想再了解可以看下开头提到的三篇文章

看到文章的最好进博客看文章哦，体验应该是最好的