K8s 初见

CreatedBy: 彭玲 CreatedOn: 2021/8/31

虚拟化

产生背景

软件开发最大的麻烦事之一，就是**环境配置**。用户计算机的环境都不相同，你怎么知道自家的软件，能在哪些机器上跑起来?

开发者常常会说 "它在我的机器可以跑的"，言下之意，其他机器很可能跑不了。

因此，软件使用必须保证两件事：操作系统的设置，各种库和组件的安装。只有它们都正确，软件才能运行。例如，安装一个 Python 应用，计算机必须有 Python 引擎，还必须有各种依赖，可能还要配置环境变量。

环境配置如此麻烦，换一台机器，就要重来一次，能不能从根本上解决问题，软件可以带环境安装？也就是说，安装的时候，把原始环境一模一样地复制过来。

解决方案

两个核心的虚拟化技术：虚拟机（系统级）、容器（进程级）。

虚拟机 (virtual machine) 就是带环境安装的一种解决方案。用户可以通过虚拟机还原软件的原始环境。但虚拟机是操作系统级别的，比较重。

容器 (container) 不是模拟一个完整的操作系统，而是对进程进行隔离。相比虚拟机，容器具有启动快、体积小、占用资源少等优点。

Docker 和 编排

Docker 属于 Linux 容器的一种封装，提供简单易用的容器使用接口，是目前最流行的 Linux 容器解决方案。

编排 (orchestration) 是对计算机系统和软件的自动化配置、协调和管理。Docker 使用 docker-compose 进行容器编排，但使用上较复杂，不适合生产级别的容器部署。

K8s 是什么?

Kubernetes 这个名字源于希腊语，意为"舵手"。k8s 这个缩写是因为 k 和 s 之间有 8 个字符。

K8s 是 Google 在 2014 年开源的一个可移植的、可扩展的开源平台，用于管理容器化的工作负载和服务。

为什么用 K8s?

在生产环境中，你需要管理运行应用程序的容器，并确保不会停机。 例如，如果一个容器发生故障，则需要启动另一个容器。如果系统处理此行为，会不会更容易？这就是 Kubernetes 解决这些问题的方法，K8s 提供了一个可弹性运行分布式系统的框架，满足你的扩展要求、故障转移、部署模式等。

此外，K8s 不仅仅是生产级别的容器编排系统，它还拥有一个庞大且快速增长的生态系统。

K8s 提供哪些功能?

- **服务发现和负载均衡**

  Kubernetes 可以使用 DNS 名称或自己的 IP 地址公开容器，如果进入容器的流量很大，Kubernetes 可以负载均衡并分配网络流量，从而使部署稳定。

- **存储编排**

  Kubernetes 允许你自动挂载你选择的存储系统，例如本地存储、公共云提供商等。

- **自动部署和回滚**

  你可以使用 Kubernetes 描述已部署容器的所需状态，它可以以受控的速率将实际状态 更改为期望状态。例如，你可以自动化 Kubernetes 来为你的部署创建新容器， 删除现有容器并将它们的所有资源用于新容器。

- **自动完成装箱计算**

  Kubernetes 允许你指定每个容器所需 CPU 和内存（RAM）。 当容器指定了资源请求时，Kubernetes 可以做出更好的决策来管理容器的资源。

- **自我修复**

  Kubernetes 重新启动失败的容器、替换容器、杀死不响应用户定义的 运行状况检查的容器，并且在准备好服务之前不将其通告给客户端。

- **密钥与配置管理**

  Kubernetes 允许你存储和管理敏感信息，例如密码、OAuth 令牌和 ssh 密钥。 你可以在不重建容器镜像的情况下部署和更新密钥和应用程序配置，也无需在堆栈配置中暴露密钥。

如何使用 K8s 部署应用?

以智慧小区 web-api 部署为例。

1. 创建部署文件

两个 yaml 部署文件：deployment.yml 和 service.yml 。

- deployment.yml

```
apiVersion: apps/v1 # for versions before 1.8.0 use apps/v1beta1
kind: Deployment
metadata:
  name: household-webapi
  namespace: smartapp
spec:
  selector:
    matchLabels:
      app: household-webapi-deployment-template-label
  replicas: 1
  template:
    metadata:
      labels:
        app: household-webapi-deployment-template-label
    spec:
      containers:
      - name: household-webapi-deployment-container
        image: repository.anxinyun.cn/smartcity/household.webapi:dragon.40
        command: [ "node", "server.js"]
        args: [ "-g",
"postgres://FashionAdmin:123456@10.8.30.39:5432/SmartHousehood", "-a",
"http://10.8.30.157:19084"]
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
        - containerPort: 8080
        volumeMounts:
        - name: hostfile
          mountPath: /etc/hosts
        - name: localtime
          mountPath: /etc/localtime
      volumes:
      - name: hostfile
        hostPath:
          path: /etc/hosts
      - name: localtime
        hostPath:
          path: /etc/localtime
```

- service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: household-webapi
  namespace: smartapp
spec:
  type: ClusterIP
  ports:
```

```
  - port: 8087
    targetPort: 8080
  selector:
    app: household-webapi-deployment-template-label
  externalIPs:
  - 10.8.30.37
```

2. K8s 部署服务

在 deployment.yml 和 service.yml 目录下执行：

```
$ kubectl apply -f .
```

3. 验证部署

```
# 查看 household-webapi
$ kubectl get pods -n smartapp | grep household-webapi
anxin@node38:~$ kubectl get pod -n smartapp
NAME                          READY       STATUS    RESTARTS       AGE
household-webapi-78c6b864d5-tfdld       1/1      Running    0          182d

# 查看 household-webapi 服务
anxin@node38:~$ kubectl get services -n smartapp | grep household-webapi
NAME           TYPE        CLUSTER-IP     EXTERNAL-IP     PORT(S)     AGE
household-webapi    ClusterIP     10.1.72.199     10.8.30.37     8087/TCP    370d
```