Kubernetes 101/201

张鑫 才云科技

主要内容

基本概念

架构设计及原理

工作流程

资源模型

网络模型与工具

存储模型

Kubernetes basic concepts

Containers are great, but let’s be honest, they only solve part of the problem while creating new ones. Focus on the Application.

-- Kelsey Hightower

- 任务调度
- 健康检查
- 故障应对
- 监测预警
- 弹性伸缩
- 服务发现

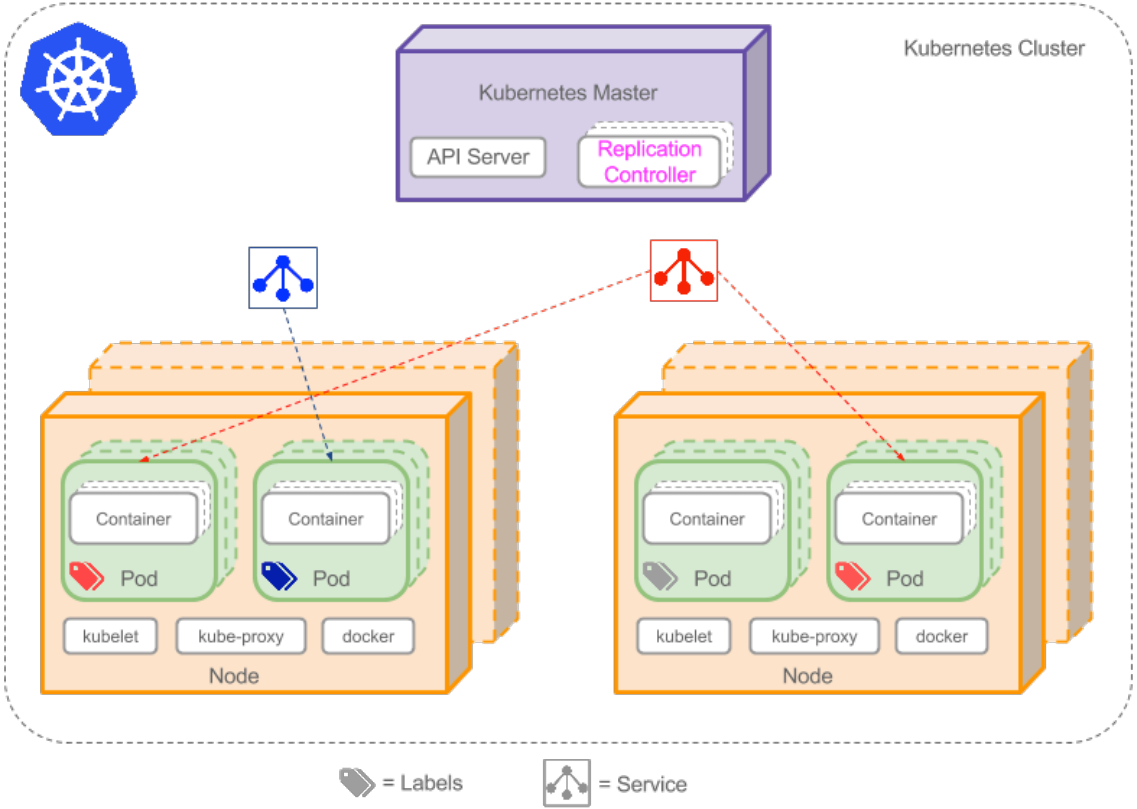
什么是 kubernetes

Kubernetes (k8s) 是 Google 开源的容器集群管理系统，前身是 Borg，用于容器的部署、自动化调度和扩展机制等。

主要特性：

- 容器的自动化部署
- 自动化扩展或者缩容
- 容器可成组，对外提供服务，支持负载均衡
- 轻松实现应用镜像升级
- 容器的健康检查，自动重启

集群



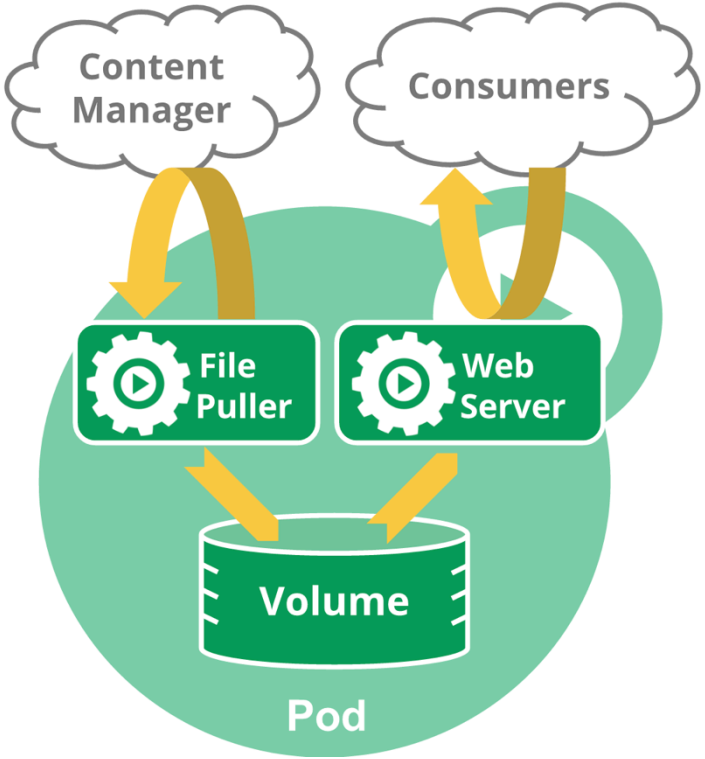
节点（Node）

- 物理机/虚拟机
- Master（管理节点）
  - 资源管理
  - Pod 调度
  - 弹性伸缩
  - 安全控制
  - 系统监控
  - 纠错
- Node（工作节点）

```
root@caicloud:/home/vagrant# kubectl get nodes
NAME            STATUS                     AGE
kube-master-1   Ready,SchedulingDisabled   3d
kube-node-1     Ready                      3d
root@caicloud:/home/vagrant#
```

Pod

- 多个容器的集合
- 最小调度单位
- 共享 namespace
  - PID 名字空间（同一 Pod 内的容器能看到其他容器的 PID，Docker 容器暂不支持）
  - 网络名字空间（共享 IP 和 localhost）
  - IPC 名字空间（能够使用 SystemV IPC 或者POSIX 消息队列进行通信）
  - UTS 名字空间（共享同一个主机名）
- 共享 Volumes

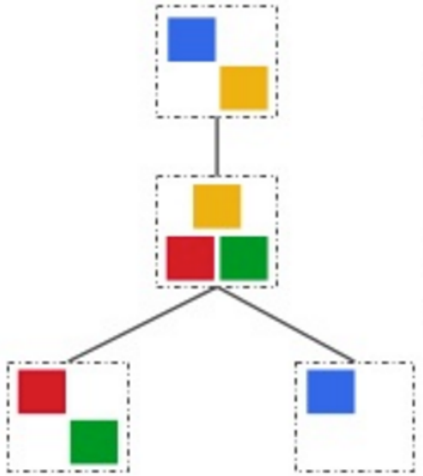


Pod cont’d

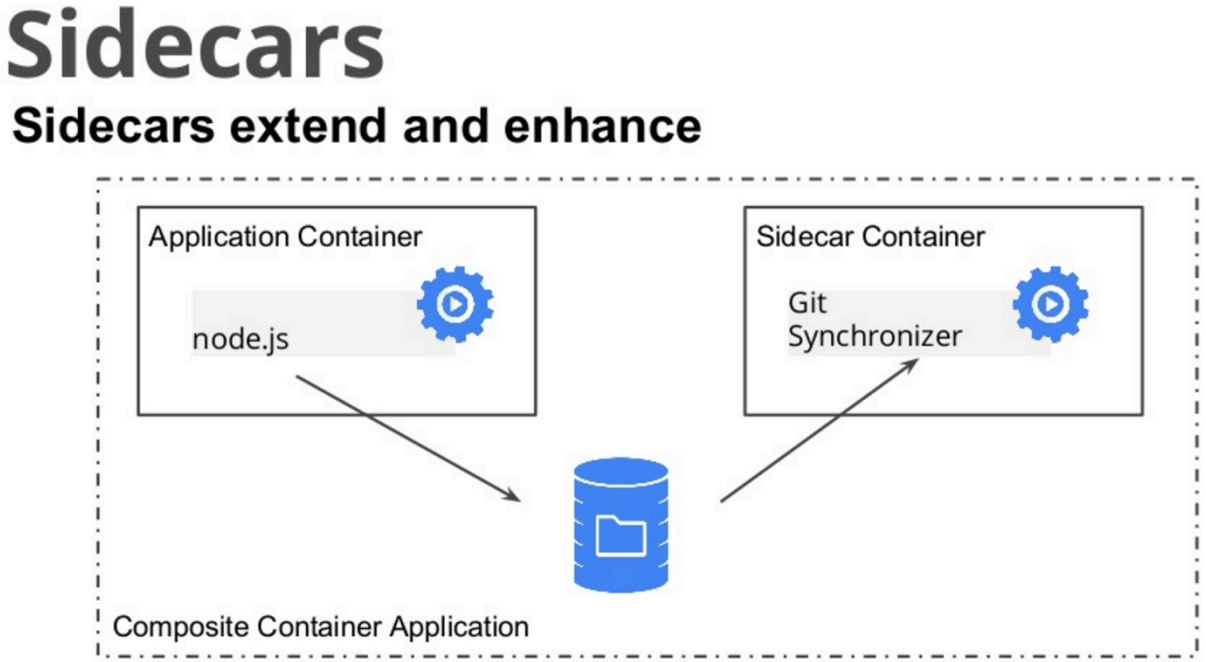
- Pod 的网络原则：— Pod — IP
  - 防止端口冲突
  - 方便服务注册
- Pod 的生命周期
  - PENDING, RUNNING, FAILED, SUCCEEDED, UNKNOWN
  - 重启策略: Always, Never, OnFailure
  - PreStop and PostStart Hooks
- Security Policies
  - 运行 Pod 化的服务

Why Pod? Design Patterns!

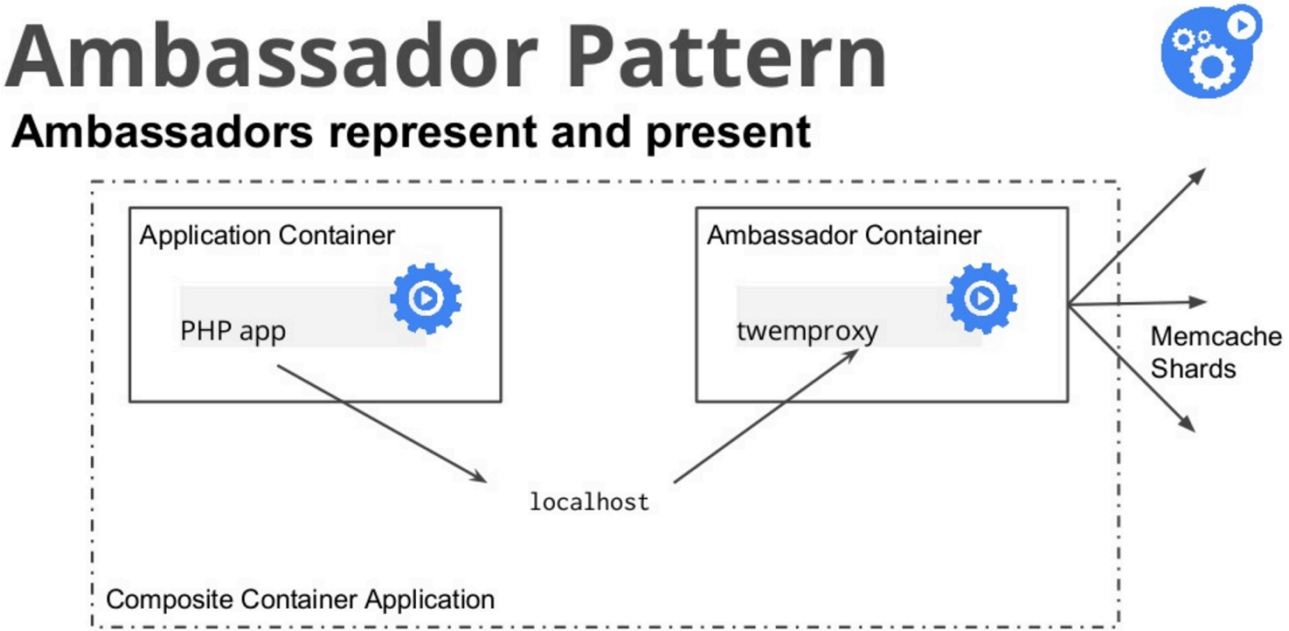
- ❏ Good fences make good neighbors!
- ❏ Composing containers for modular architectures
- ❏ Containers are more like (ation boundaries
- ❏ Design Patterns!



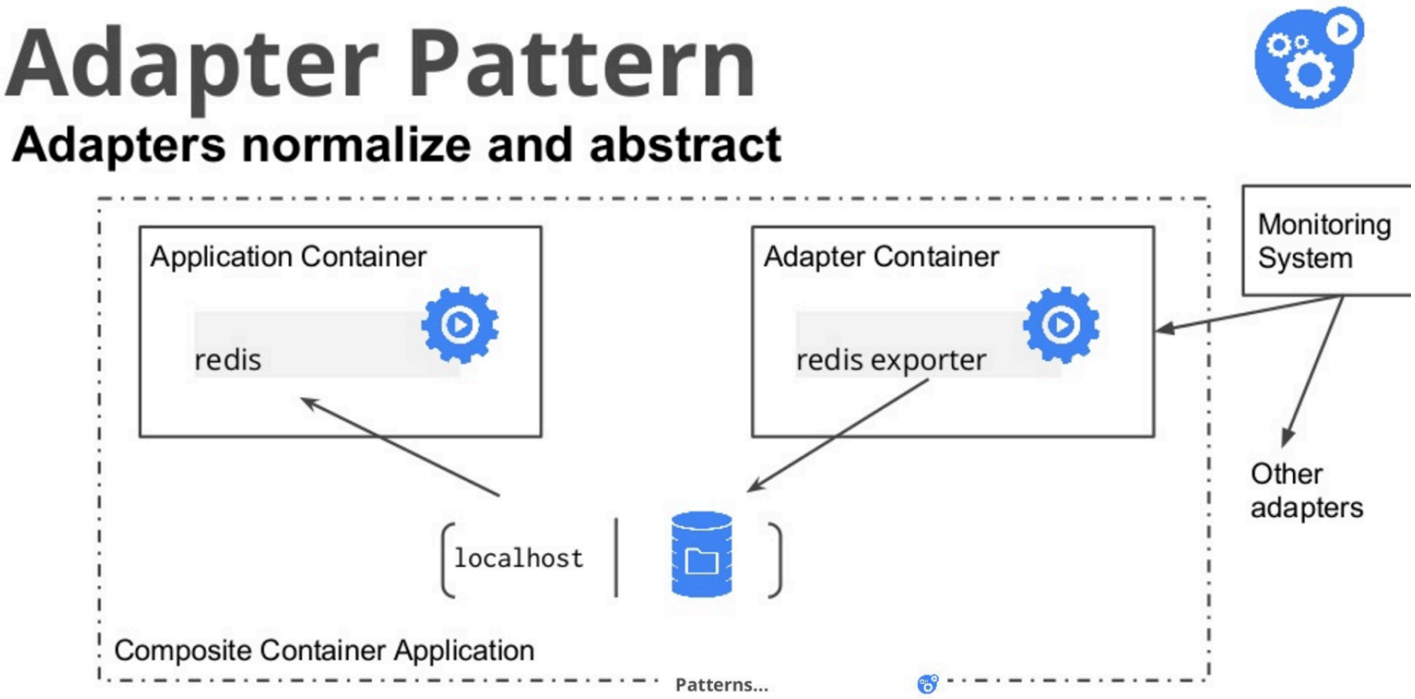
Why Pod?



Why Pod?



Why Pod?



Label 和 Selector

- label
  - 用于区分资源的 key/value 键值对
  - 资源可以拥有多个 label

```
"labels": {
    "environment": "production",
    "tier": "frontend",
    "component": "redis"
}
```

- selector
  - 根据 label 选择对应的资源
  - Equality-based 选择器
    - kubectl get pods -l environment=production,tier=frontend
  - Set-based 选择器
    - kubectl get pods -l 'environment in (production),tier in (frontend)'

Deployment

- 确保任意时间都有指定数量的 Pod “副本”在运行
- 通过 Pod 选择器筛选出对应的 Pod 实例并实时监控其状态和数量

副本数

Pod 选择器

Pod 标签

```
apiVersion: v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Deployment

- 提供 Pod 和 Replica Set 的声明式更新
- 把部署，滚动更新和回滚进行了自动化
- 创建 Deployment
  - kubectl create -f docs/user-guide/nginx-deployment.yaml --record
- 更新 Deployment
  - kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
- 查看 Deployment 的修订版本
  - kubectl rollout history deployment/nginx-deployment

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

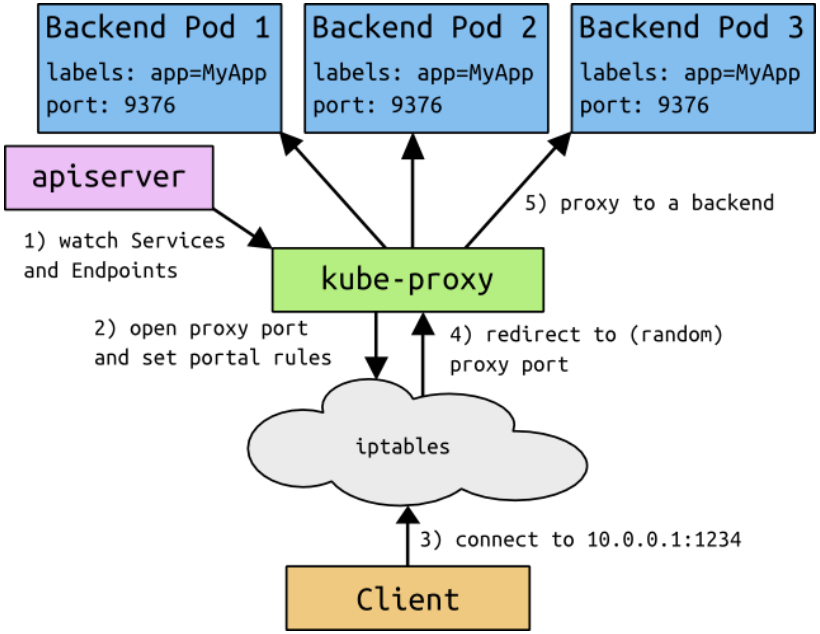
Service

- 一个唯一指定的名字
- 一个虚拟 IP 和端口号
- 服务类型
  - ClusterIP（default）
  - Nodeport
  - LoadBalancer
- 服务发现
  - 环境变量
  - DNS

```
{
   "kind": "Service",
   "apiVersion": "v1",
   "metadata": {
      "name": "my-service"
   },
   "spec": {
      "selector": {
         "app": "MyApp"
      },
      "ports": [
         {
            "protocol": "TCP",
            "port": 1234,
            "targetPort": 9376,
            "nodePort": 30029
         }
      ],
      "type": "NodePort"
   }
}
```

ClusterIP:port

NodeIP:nodePort



Revisit：Label 的用途

- 资源分组和过滤
  - kubectl get pods -lapp=guestbook,role=slave
- 批量操作
  - kubectl delete deployment,services -l app=nginx

```
name: frontend
replicas: 3
...
labels:
   app: guestbook
   tier: frontend
   track: stable
...
image: gb-frontend:v3
```

```
name: frontend-canary
replicas: 1
...
labels:
   app: guestbook
   tier: frontend
   track: canary
...
image: gb-frontend:v4
```

```
selector:
   app: guestbook
   tier: frontend
```

Revisit: 资源关系

Basic Unit: Pod, Node, Volume, Label, Endpoint, Binding, etc

Aggregation: ReplicaSet, DaemonSet, StatefulSet, Deployment, PersistentVolume, etc

Control loop: kube-proxy, scheduler, deployment controller, etc