Kubernetes 架构原则和对象设计

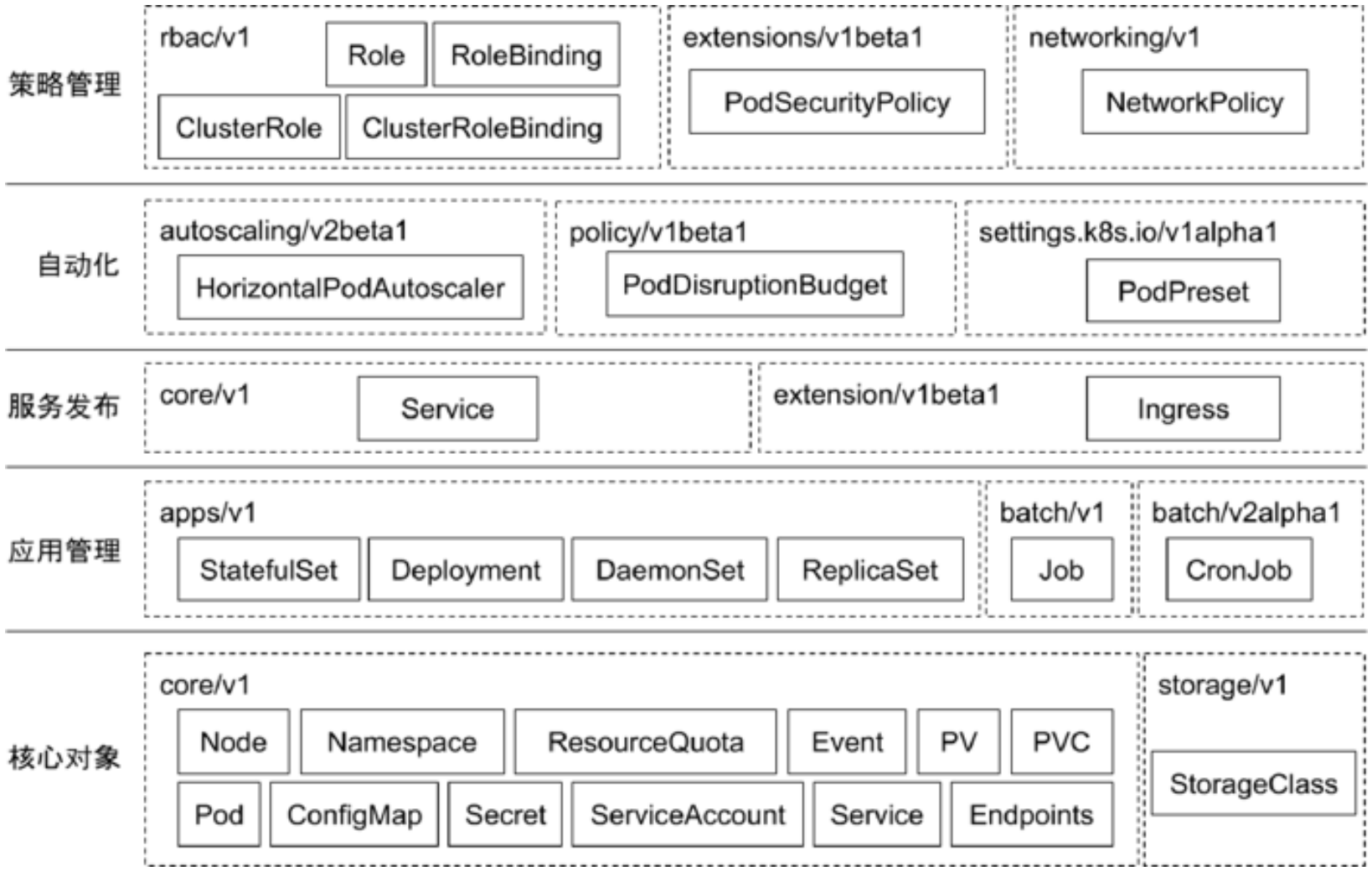
孟凡杰

腾讯云容器技术专家

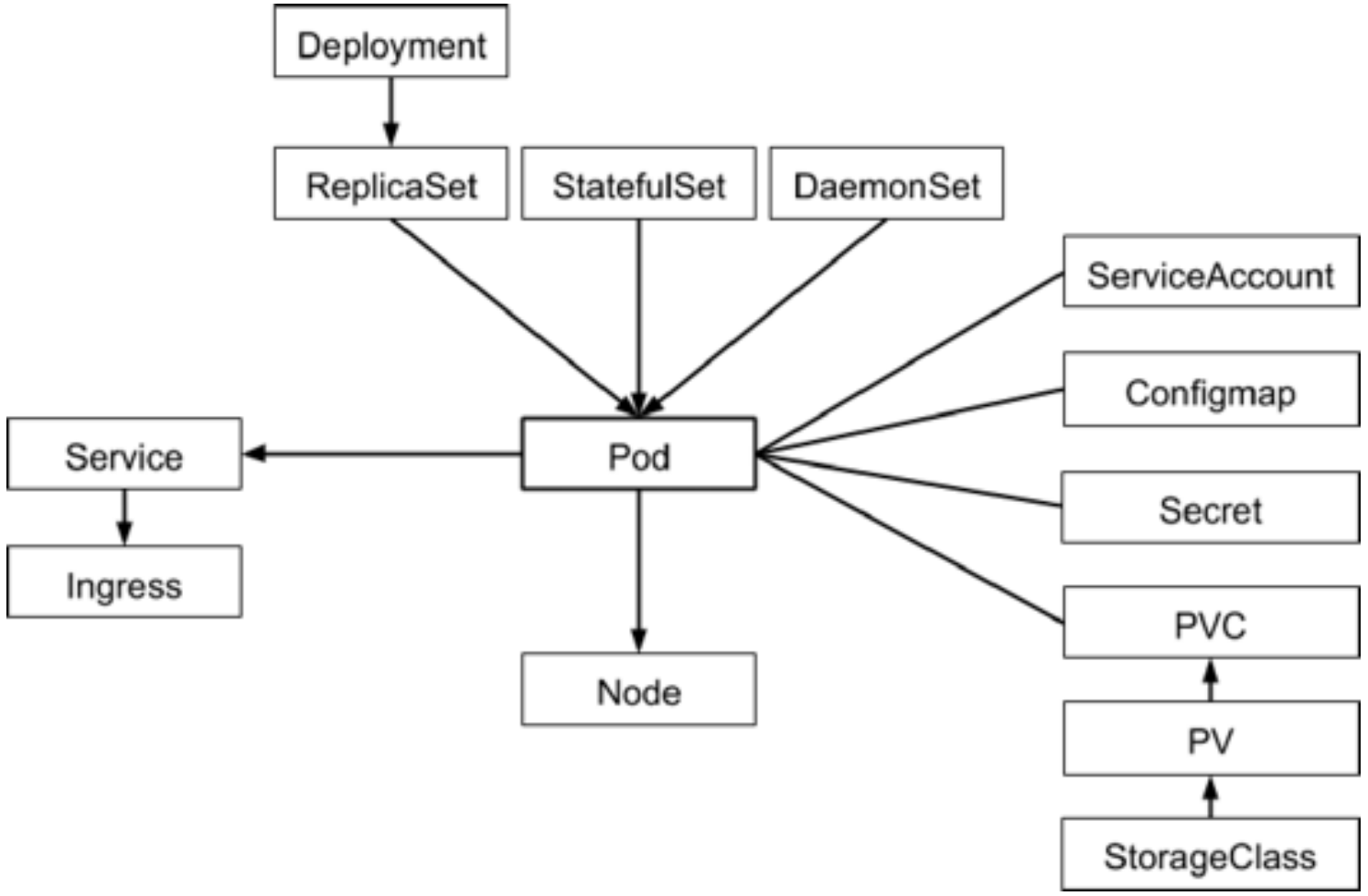
核心技术概念和 API 对象

- API 对象是 Kubernetes 集群中的管理操作单元；
- Kubernetes 集群系统每支持一项新功能，引入一项新技术，一定会新引入对应的 API 对象，支持对该功能的管理操作；
- 例如副本集 Replica Set 对应的 API 对象是 RS；
- 深入理解 Kubernetes 对象的通用设计：
    - TypeMeta
        - G(roup)
        - K(ind)
        - V(ersion)
    - Metadata
        - Namespace
        - Name
        - Labels & Annotation
        - Finalizers
        - ResourceVersion
        - SelftLink

常用 Kubernetes 对象及其分组



核心对象间的关系图


Deployment
ReplicaSet
StatefulSet
DaemonSet
ServiceAccount
Configmap
Service
Pod
Secret
Ingress
PVC
Node
PV
StorageClass

Node

- Node 是 Pod 真正运行的主机，可以物理机，也可以是虚拟机。
- Node 对象用来描述节点的计算资源：
    - Capacity：计算能力，代表当前节点的所有资源；
    - Allocatable：可分配资源，代表当前节点的可分配资源，通常是所有资源-预留资源-已分配资源；
- Kubelet 会按固定频率检查节点健康状态并上报给 APIServer，该状态会记录在 Node 对象的 status 中。

Pod

- Pod 是一组紧密关联的容器集合，是 Kubernetes 调度的基本单位，容器进程运行在不同 PID、IPC、Network 和 UTS namespace，且彼此之间共享网络。
- Pod 的设计理念是支持多个容器在一个 Pod 中共享网络和文件系统，可以通过进程间通信和文件共享这种简单高效的方式组合完成服务。

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

副本集（ReplicaSet）

- Pod 描述的是具体的应用实例，当 Pod 被删除后，就彻底消失了；
- 为保证应用的高可用，引入副本集来确保应用的总副本数永远与期望一致；
- 若某个 Pod 隶属于某个副本集，若该 Pod 被删除，则 ReplicaSet Controller 会发现当前运行的副本数量与用户的期望不一致，则会创建新的 Pod。

部署（Deployment）

- 部署表示用户对 Kubernetes 集群的一次更新操作。
- 部署是一个比 RS 应用模式更广的 API 对象，可以是创建一个新的应用，更新一个已存在的应用，也可以是滚动升级一个应用。
- 滚动升级一个服务，实际是创建一个新的 RS，然后逐渐将新 RS 中副本数增加到理想状态，将旧 RS 中的副本数减小到 0 的复合操作；这样一个复合操作用一个 RS 是不太好描述的，所以用一个更通用的 Deployment 来描述。以 Kubernetes 的发展方向，未来对所有长期伺服型的的业务的管理，都会通过 Deployment 来管理。

服务（Service）

RC、RS 和 Deployment 只是保证了支撑服务的微服务 Pod 的数量，但是没有解决如何访问这些服务的问题。

一个 Pod 只是一个运行服务的实例，随时可能在一个节点上停止，在另一个节点以一个新的 IP 启动一个新的 Pod，因此不能以确定的 IP 和端口号提供服务。要稳定地提供服务需要服务发现和负载均衡能力。服务发现完成的工作，是针对客户端访问的服务，找到对应的的后端服务实例。

在 Kubernetes 集群中，客户端需要访问的服务就是 Service 对象。每个 Service 会对应一个集群内部有效的虚拟 IP，集群内部通过虚拟 IP 访问一个服务。

在 Kubernetes 集群中微服务的负载均衡是由 kube-proxy 实现的。kube-proxy 是 Kubernetes 集群内部的负载均衡器。它是一个分布式代理服务器，在 Kubernetes 的每个节点上都有一个；这一设计体现了它的伸缩性优势，需要访问服务的节点越多，提供负载均衡能力的 kube-proxy 就越多，高可用节点也随之增多。

与之相比，我们平时在服务器端使用反向代理作负载均衡，还要进一步解决反向代理的高可用问题。

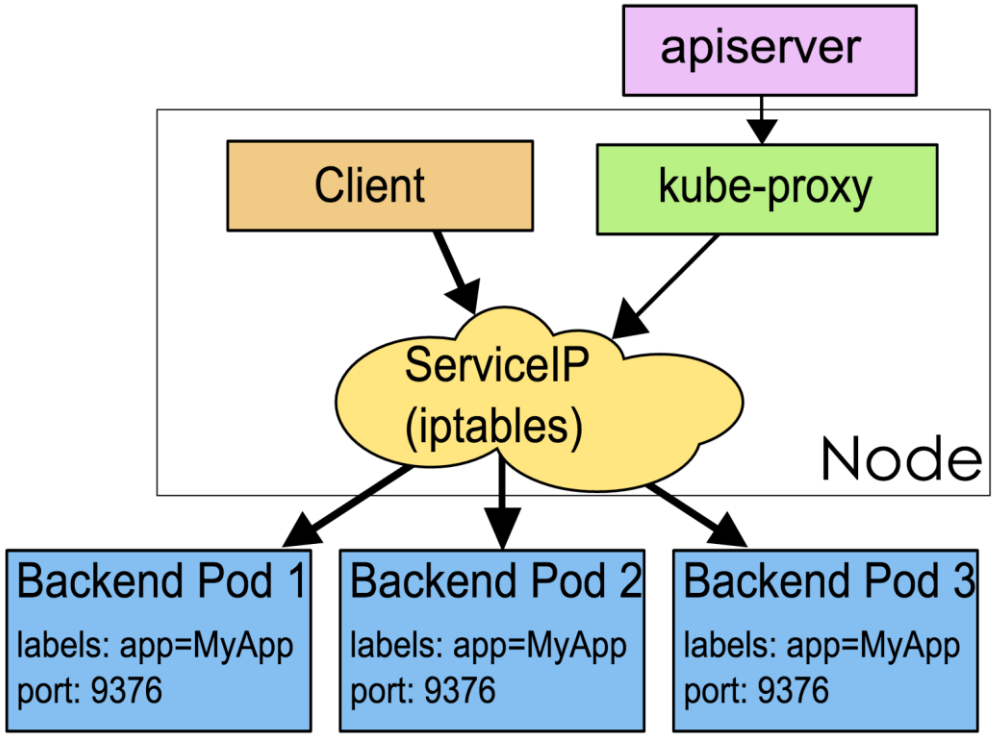
Namespace

- Namespace 是对一组资源和对象的抽象集合，比如可以用来将系统内部的对象划分为不同的项目组或用户组。
- 从 Namespace 划分的维度看，Kubernetes 对象分为两类：
  - Non-namespaced object：
    - 全局对象，这些对象不与任何 Namespace 绑定，属于集群范围的对象。如 Node， PersistVolume，ClusterRole。
  - Namespaced object：
    - 与具体 Namespace 绑定对象，如 Pod，Service。

Service

Service 是应用服务的抽象，通过 labels 为应用提供负载均衡和服务发现。匹配 labels的 Pod IP 和端口列表组成 endpoints，由 kube-proxy 负责将服务 IP 负载均衡到这些 endpoints 上。

每个默认类型 Service 都会自动分配一个 cluster IP（仅在集群内部可访问的虚拟地址）和 DNS 名，其他容器可以通过该地址或 DNS 来访问服务，而不需要了解后端容器的运行。



Service Spec

```
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  ports:
  - port: 8078 # the port that this service should serve on
    name: http
    # the container on each pod to connect to, can be a name
    # (e.g. 'www') or a number (e.g. 80)
    targetPort: 80
    protocol: TCP
  selector:
    app: nginx
```

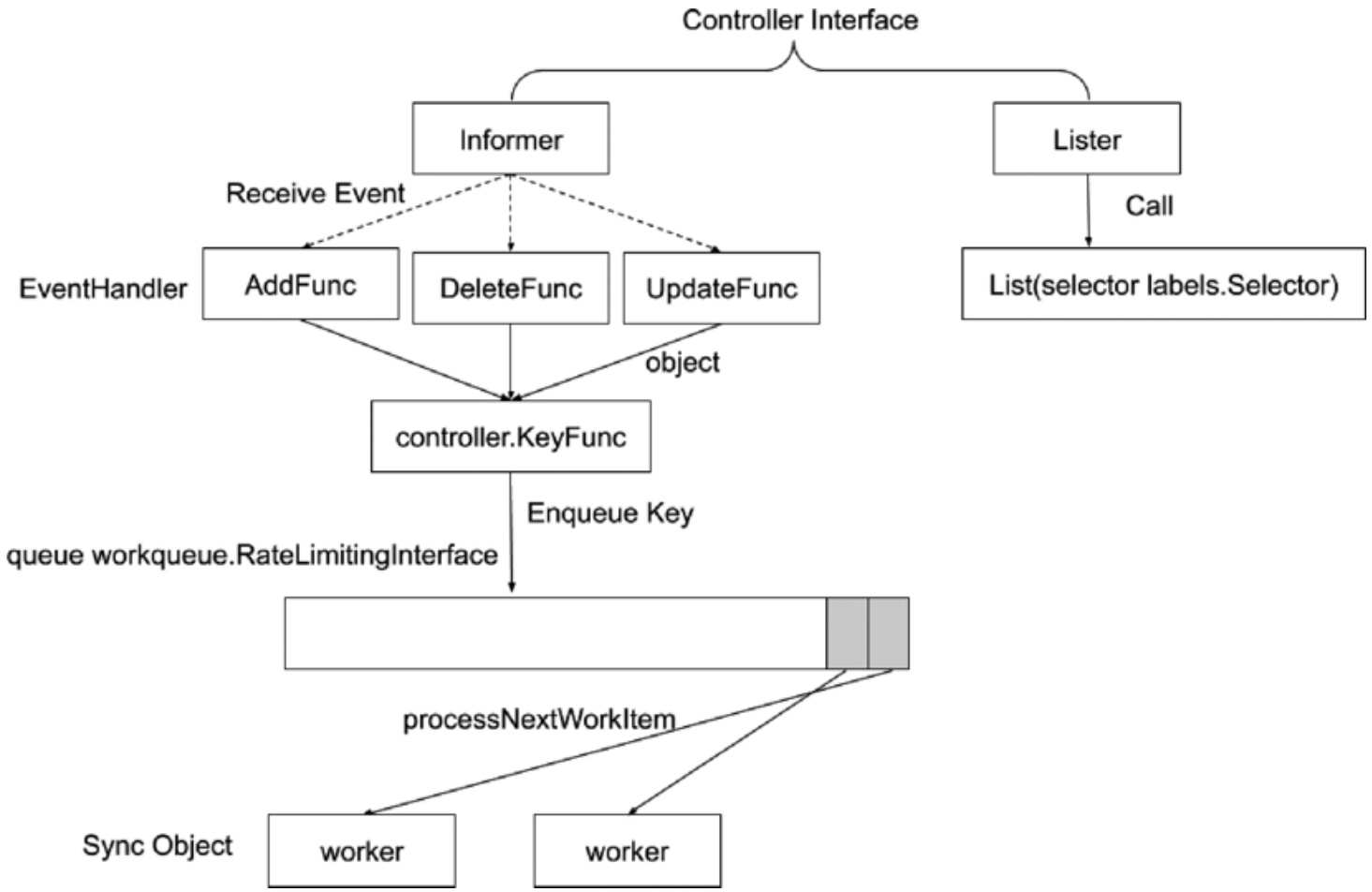
Try it

通过类似 docker run 的命令在 Kubernetes 运行容器

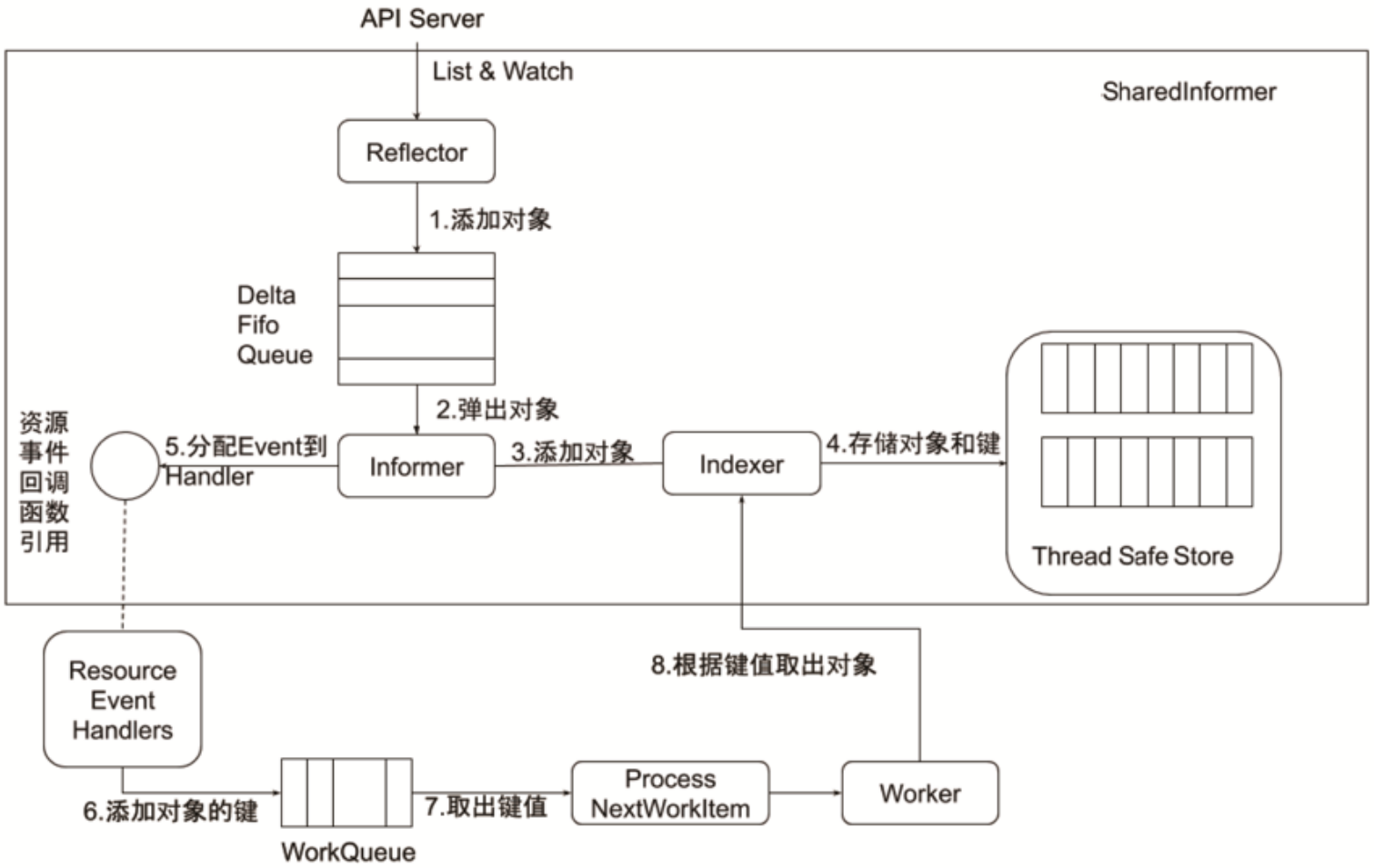
```
kubectl run --image=nginx:alpine nginx-app --port=80

kubectl get deployment

kubectl describe deployment/rs/pod
```

发生了什么？理解控制器原理

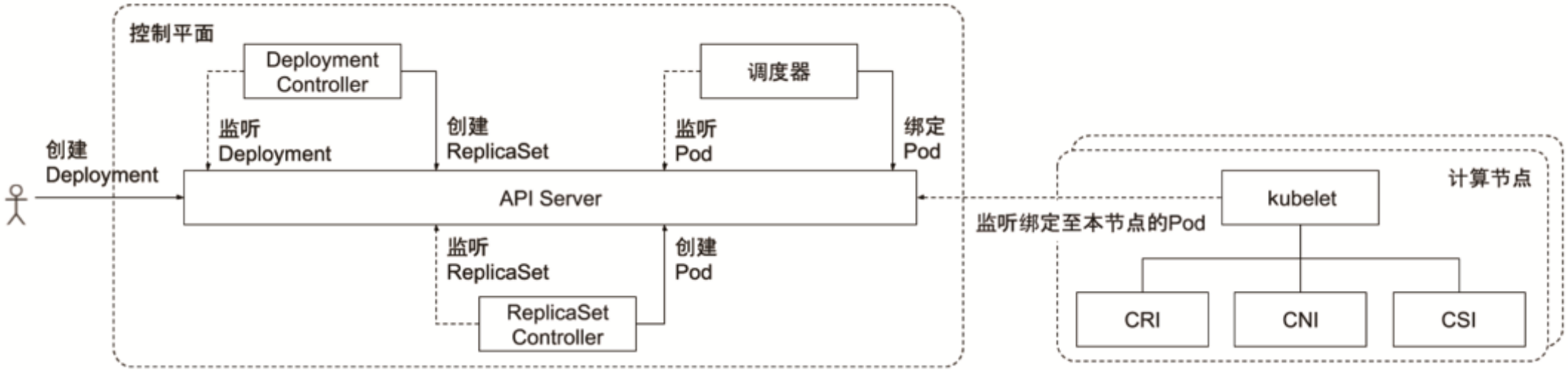
控制器的工作流程


Controller Interface
Informer
Lister
Receive Event
Call
EventHandler
AddFunc
DeleteFunc
UpdateFunc
List(selector labels.Selector)
object
controller.KeyFunc
Enqueue Key
queue workqueue.RateLimitingInterface
processNextWorkItem
Sync Object
worker
worker

Informer 的内部机制



控制器的协同工作原理


控制平面
Deployment Controller
调度器
监听 Deployment
创建 ReplicaSet
监听 Pod
绑定 Pod
创建 Deployment
API Server
监听 ReplicaSet
创建 Pod
ReplicaSet Controller
监听绑定至本节点的Pod
kubelet
计算节点
CRI
CNI
CSI

思考

思考 stateful set 对象跟 deployment 对象的不同以及不同设计的目的。

云原生训练营设计思路

- 多重视角
  - 管理员角度
    - 如何构建和运维支持生产化作业的多租户集群
    - 如何应对规模化所带来的挑战
  - 研发人员角度
    - 如何将不同类型应用的接入到容器化平台
    - 理解如何保证应用的服务可用性
  - 不同角色如何做好协同，避免出现生产故障

云原生训练营设计思路

- Go 语言基础
- 从容器技术展开，到 Kubernetes 平台以及衍生项目 Istio 等
- 理解 Kubernetes 作为开放式平台，如何与企业服务进行整合
- 如何进行多集群管理，构建两地三中心的部署模式

云原生训练营设计思路

- 编写应用
- 容器化
- 部署至 Kubernetes 集群
- 生产化运维和服务管控

学习云原生我能获得什么

- 从基础架构到应用的全方位技术提升
  - 学习微服务系统架构方法
  - 全方位学习容器技术、网络协议栈、文件系统等知识
  - 全面掌握云原生技术栈核心项目的设计原理、典型配置、常见问题以及排查手段
  - 掌握构建生产化集群的方法
  - 理解大型生产应用落地可能存在的问题和解决办法

学习云原生我能获得什么

- 流程
  - DevOps 流程
  - 如何基于 CI/CD 构建
  - 集群和应用日常运维方法
- 更多的工作机会、更资深的职位（更好的待遇）
  - 本课程对标大厂 P6-P7 的技术广度和深度
- Kubernetes 是未来不变的云计算标准
  - 技术变革从互联网行业传递到了到金融、电信、制造等传统行业

CNCF 2020 年度报告

- Kubernetes 在生产中的使用率从 78% 增加到 83% 。
- 容器在生产中的使用率从 84% 增加到 92%，超过 5000 个容器的用户在 2020 年达到 23% 。
- 69% 的招聘经理在寻找云和容器专家。

数据来源：https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf

极客时间 | 训练营

孟凡杰
腾讯云容器技术专家
前 eBay 资深架构师

极客时间 | 训练营

云原生训练营

向技术要红利，4 个月，挑战 50 万年薪

- 选择比努力更重要，云原生是新赛道
- 一线大厂都在加急招聘云原生工程师
- 懂 Kubernetes 的工程师可以弯道超车
- 简历辅导并直推知名企业

15 周	15 大	6 大	105 天	2 次
系统集训	内容模块	实践案例	助教答疑	企业内推

2022 年 9 月 19 日开课　15 周课程 | 开课 7 天内可退款 | 支持花呗分期 | 可开发票

添加学习助理咨询