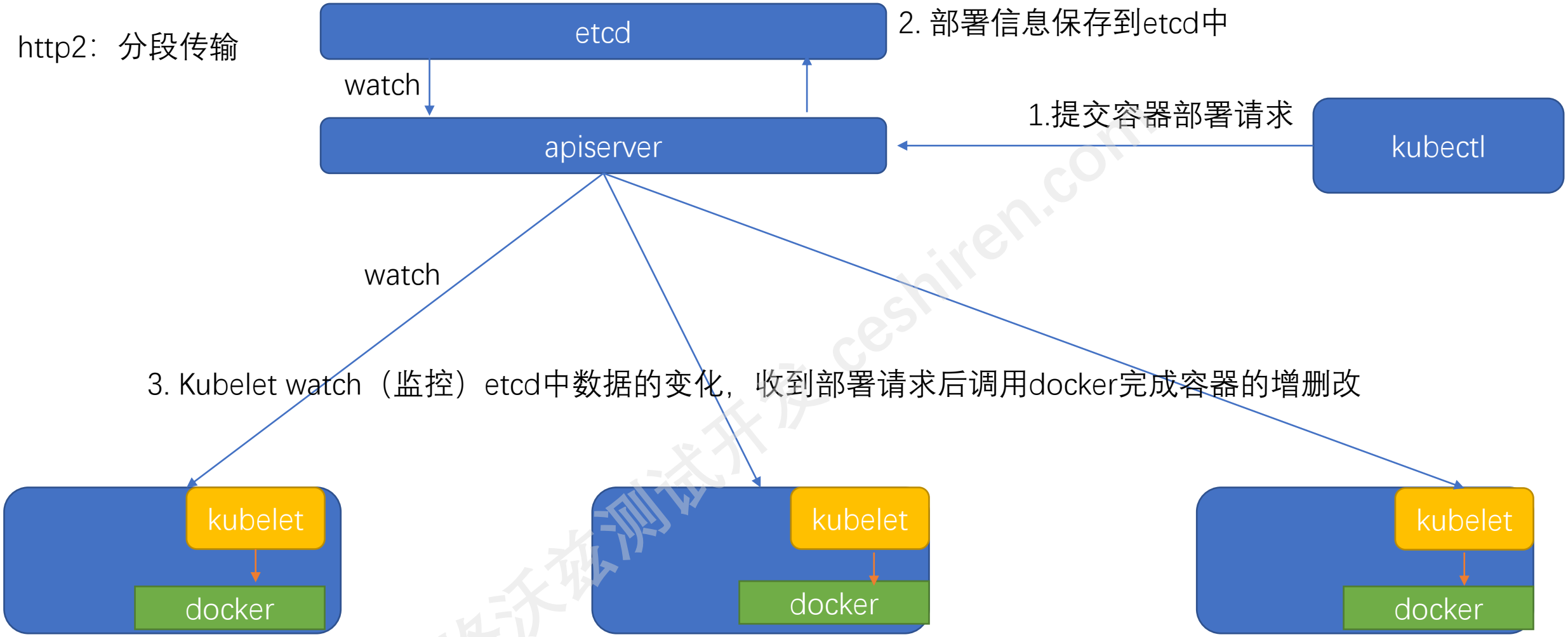
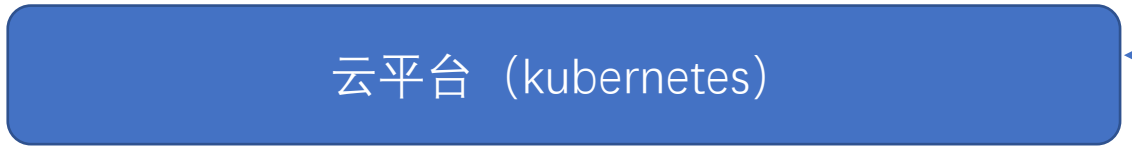
http2：分段传输



1. 通过apiserver查找当前节点的预期状态，以便操作docker执行容器的增删改。
2. 周期性的把当前节点的容器状态发送到apiserver中，并保存在etcd里。

服务与机器解耦

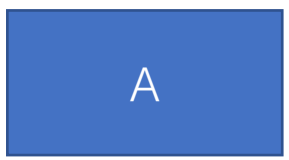
声明式API



提交

1. Cpu，内存，gpu
2. 存储（ssd，nfs，ceph）
3. 地域（城市，机房）
4. 其他调度需求

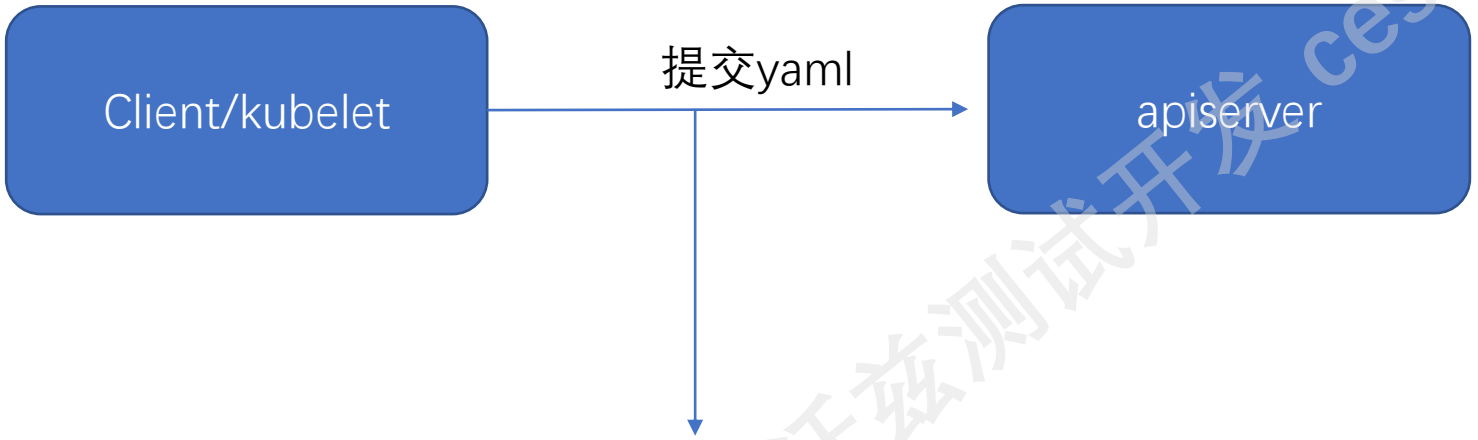
10Pod：20C 40G



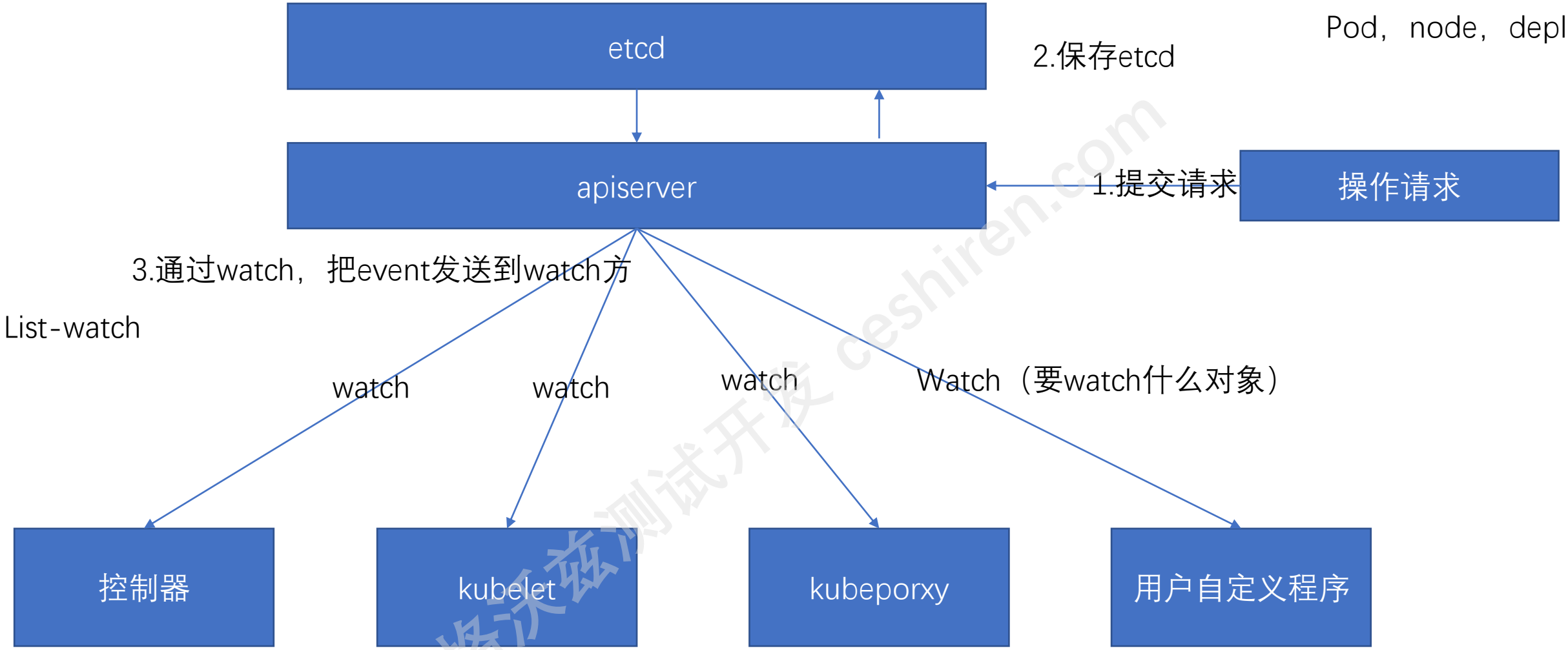
ssd: true

Ssd:true

5Pod：10C，20G



1. 服务的信息（有几个容器，元数据，env，image，容器名称。。。。）
2. 资源：cpu，内存，gpu
3. 存储：磁盘，分布式存储，等等
4. 网络策略：
5. 其他调度需求：



Pod控制，node控制器，deployment控制

4. 回调函数，处理event

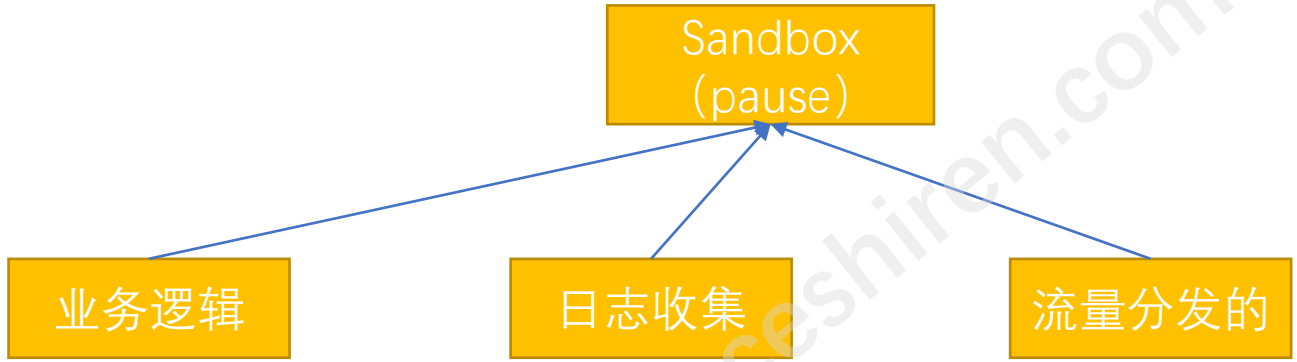
Event：

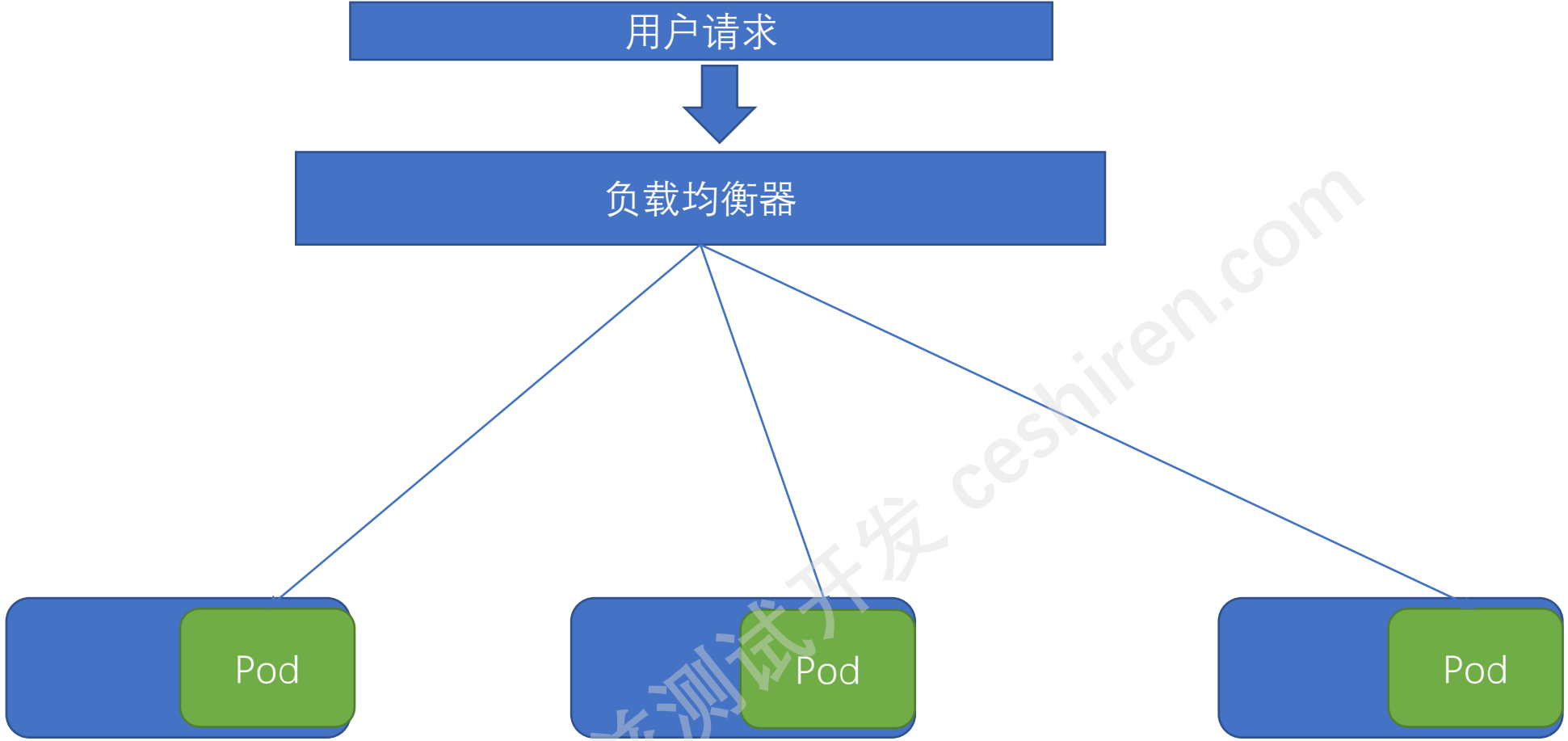
1. 事件类型（add，modify，delete）
2. 对象信息（Pod的完全定义）

控制器模式

Pod

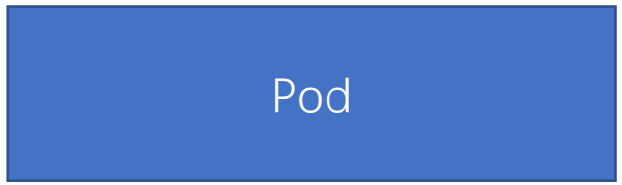
Side car

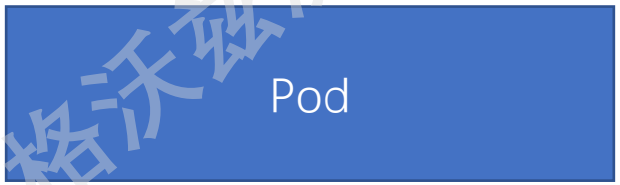


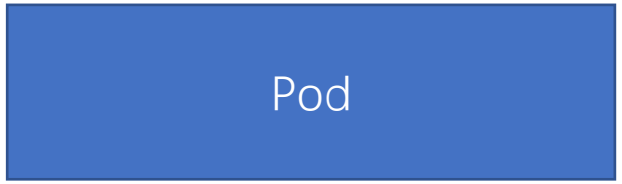


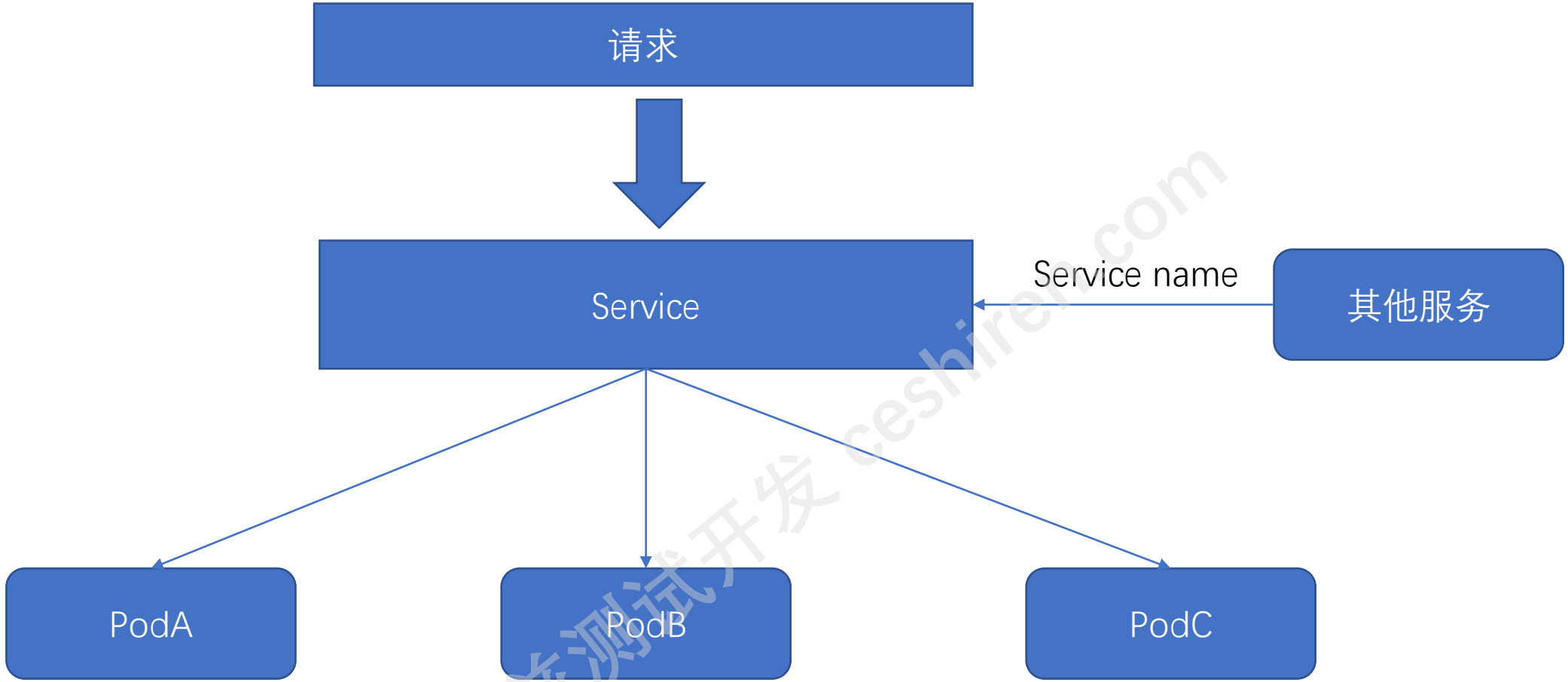
维护Pod的生存状态

Replica：3
PodTemplate：Pod定义









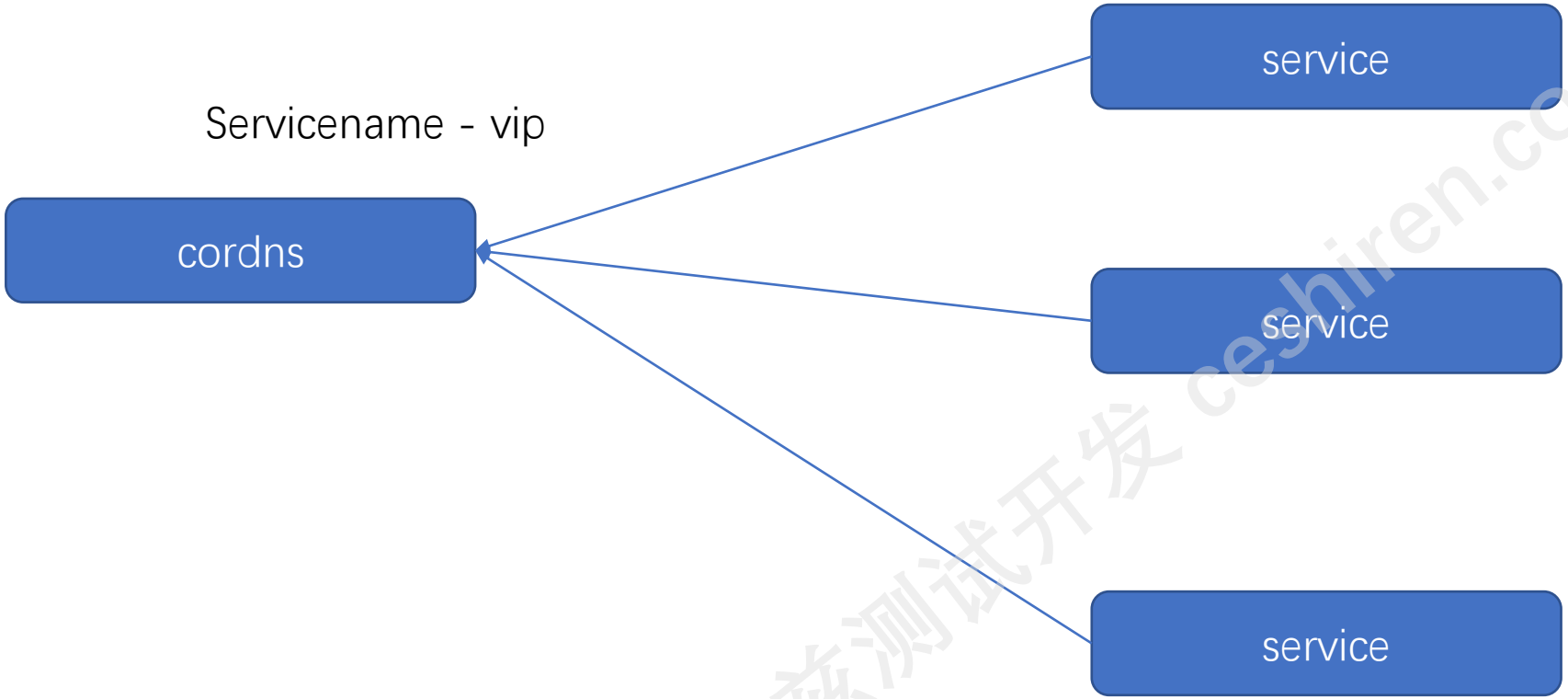
Iptables/ipvs

1. 使用iptables命令，在k8s集群中的每一个节点，创建规则：

第一条规则：请求转发到PodA的概率是33%
第二条规则：请求转发到PodB的概率是50%
第三条规则：请求转发到PodC的概率是100%

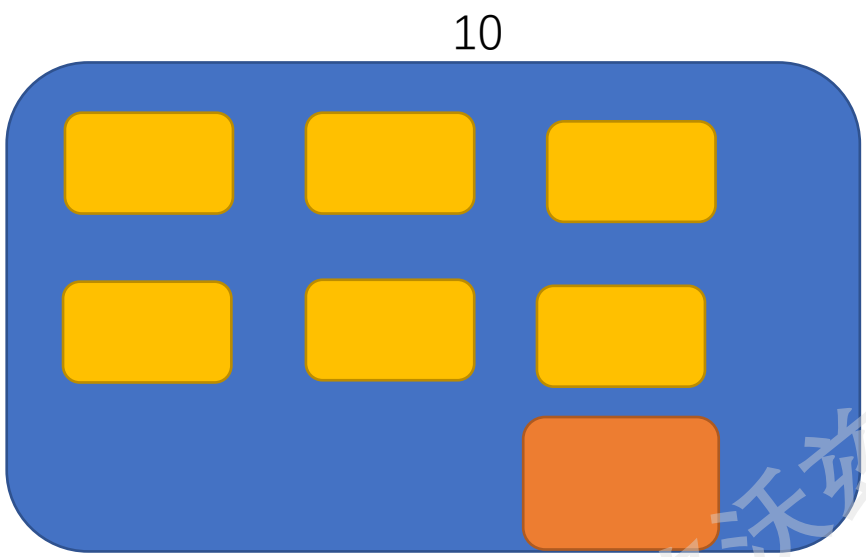
探针（健康检查）：

1. Liveness生存探针
2. Readiness 就绪探针
3. Startup 启动探针

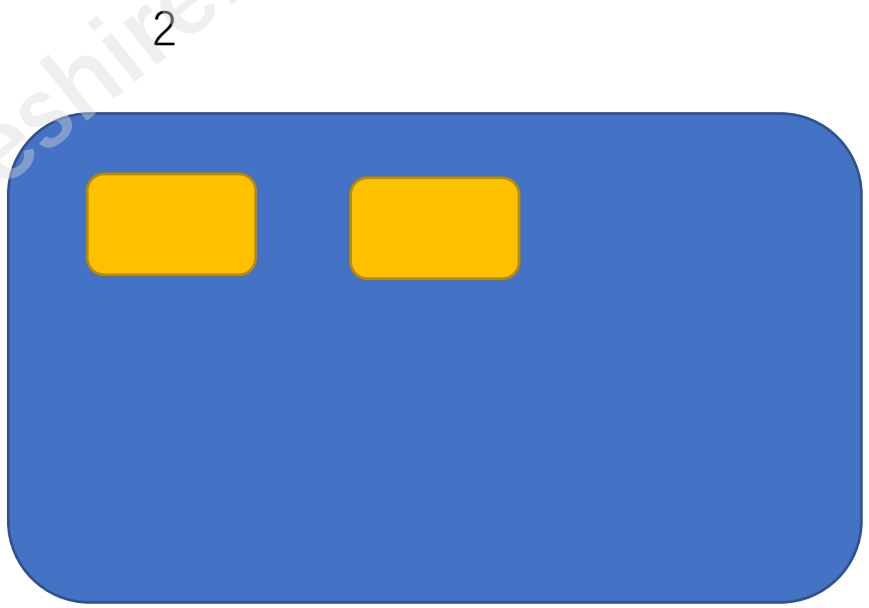


Request：k8s集群给这个容器预留的资源，即便这个容器实际上没有使用到这部分资源，K8S也会为你预留，其他容器是不能使用这部分资源的。
Limit：k8s为这个容器设置的资源上限，代表着容器最多能使用的资源，如果超过这个资源。 对于可压缩资源（cpu），K8s会对容器的cpu进行限速。对于不可压缩资源（内存）， 直接启动oom killer， 杀掉进程。

Request：1 limit： 15



Request：20



Request： 40

负载不均衡：容量测试