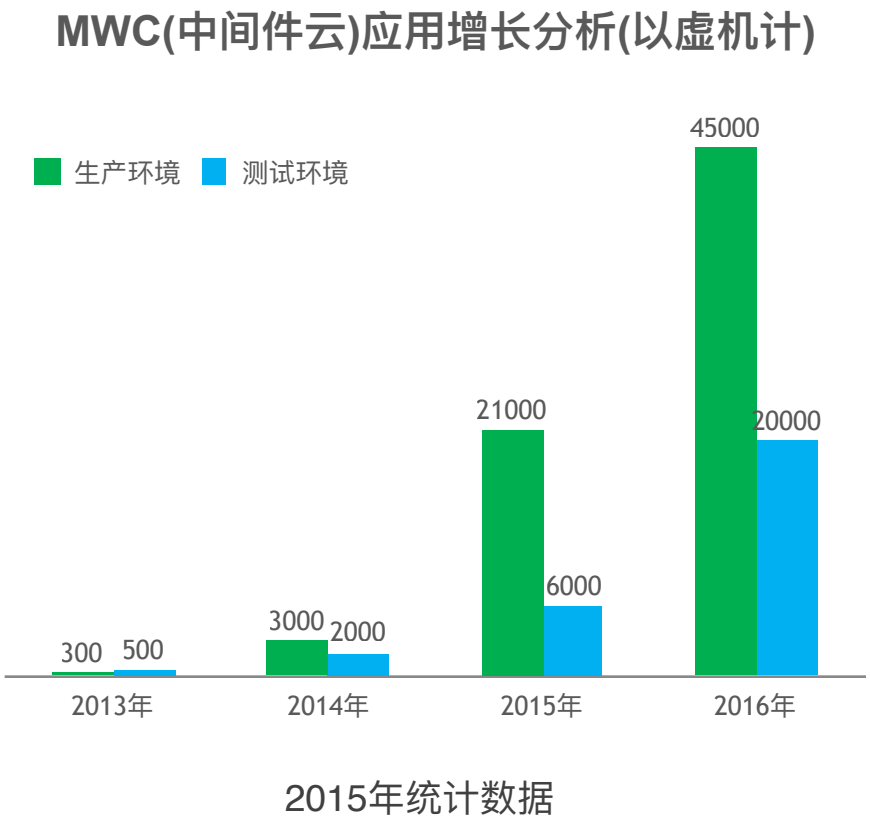
Kubernetes
在华为全球IT系统中的实践

华为IT系统业务现状



2015年统计数据

01 规模庞大 – 部署面广，覆盖全球多个地区

跨域多DC的部署维护无法拉通，运维投入巨大

02 频繁发布 – 业务繁多，迭代周期短

平台不支持快速迭代，业务上线效率低

03 冗余部署 – 同城双活、异地容灾、备份/滚动升级

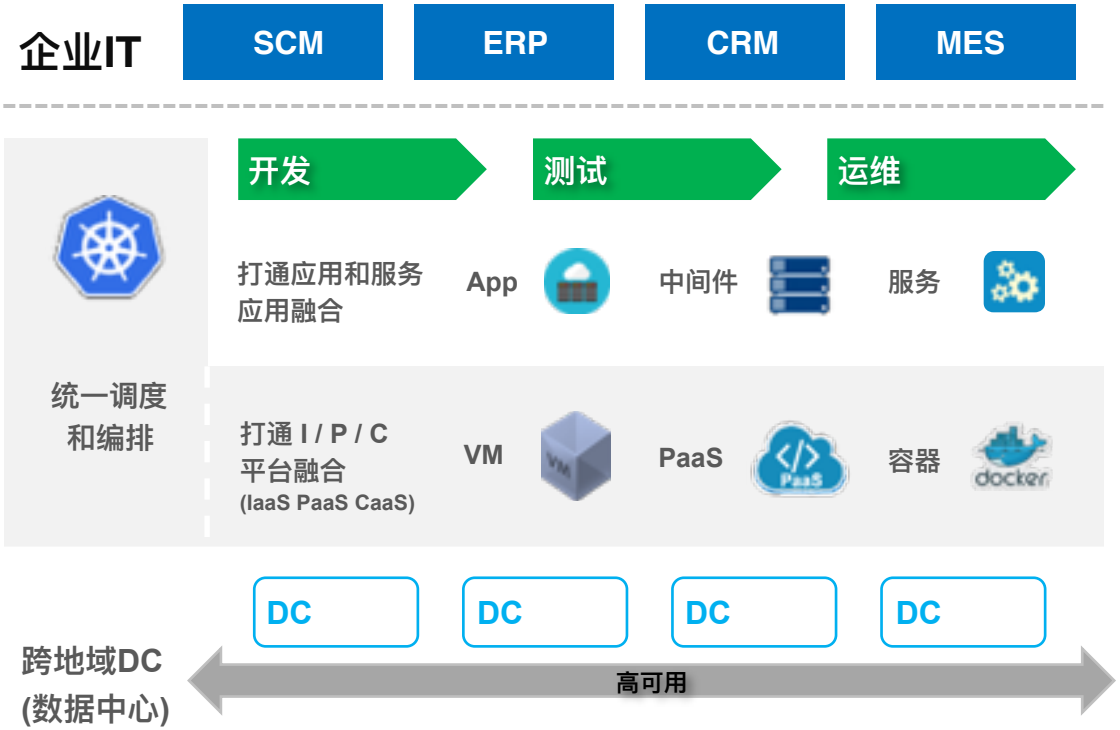
应用独占VM，资源利用率低，成本高

04 容器化、微服务化改造，应用急速增长

系统重载，无法大规模快速弹性部署

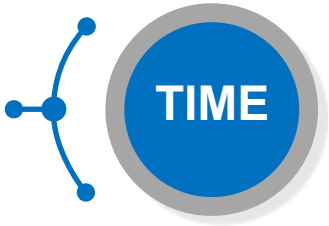
CCE在华为IT系统中的目标与定位

- 跨域应用集中部署与管理
- 混合应用资源编排
- 开发测试生产一致性环境快速供给
- 开发性快速集成传统中间件
- 支撑传统应用中间件容器化



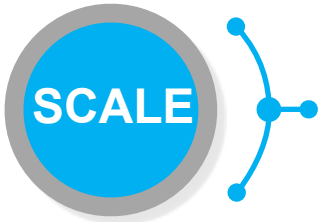
以 kuberentes、docker 为核心技术构建，融合打通原有IaaS、PaaS、CaaS平台，一个技术栈支撑多种业务场景

CCE在华为IT系统的进展



2015年底上线生产环境

- 实现自动化部署、提升资源利用率3-4倍；端到端分钟级自助在线环境获取、自动弹性伸缩，大幅加速应用上线；
- 支持华为内部IT系统多项业务从传统SOA架构向容器为中心的微服务架构平滑演进。



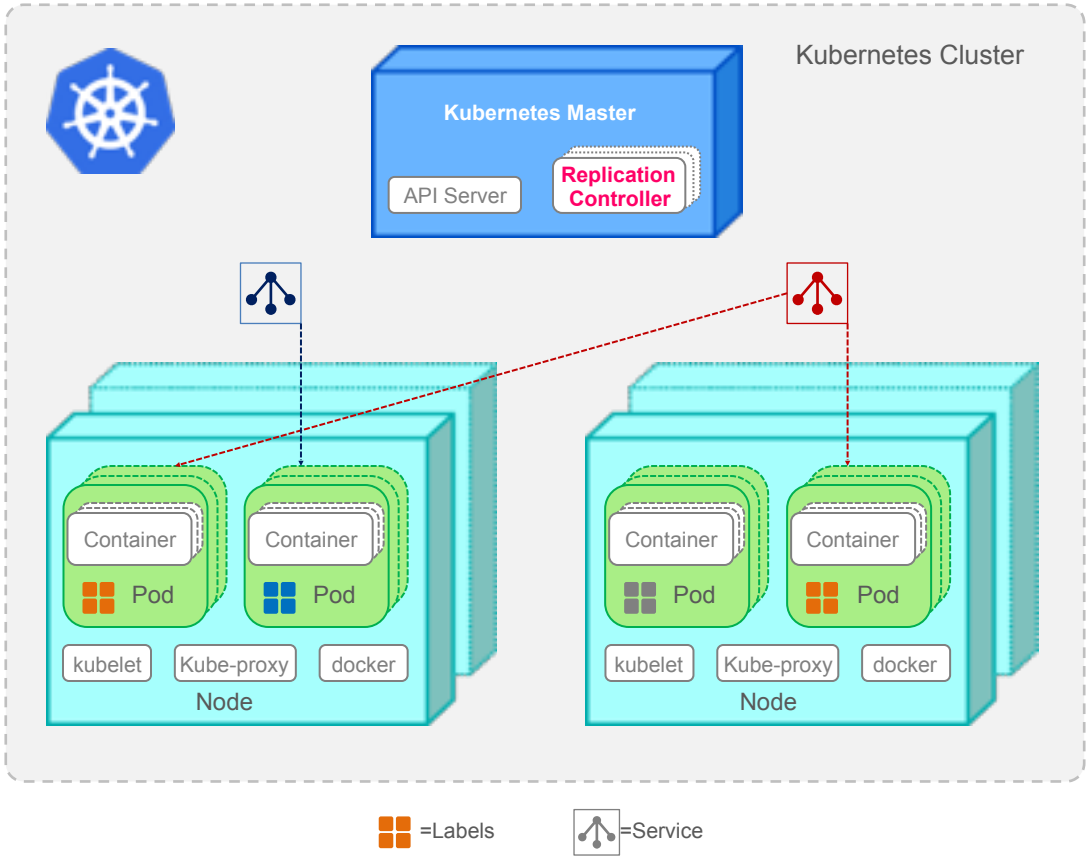
当前规模

2k+ VM，8k+ 容器

近期重点技术实践

- Kubernetes多集群联邦
- 应用间的亲和/反亲和调度

Kubernetes基本概念

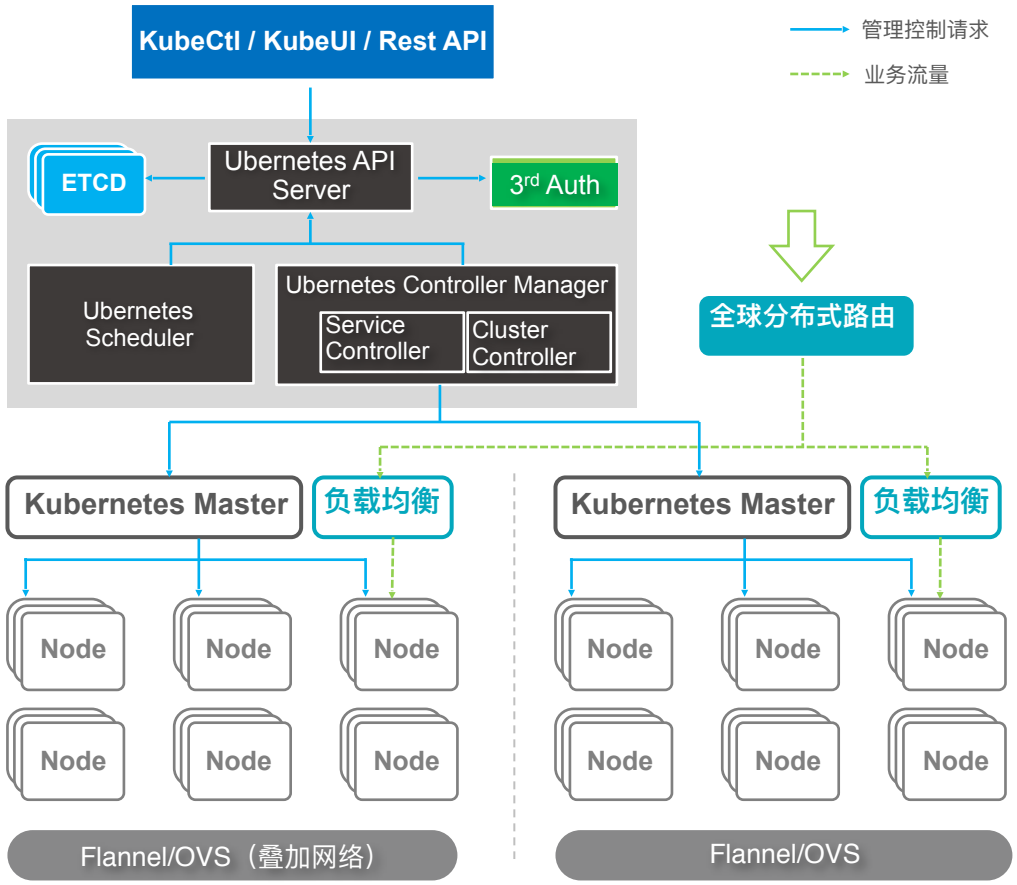


鸟语	人话
Container	容器
Pod	容器组
Replication Controller (ReplicaSet)	副本控制器 (副本集)
Service	服务
Label	标签
Node	节点
Kubelet	节点Agent
Kubernetes Master	主节点

Kubernetes集群联邦

- 多数据中心统一管理部署应用，提供**跨域的应用服务发现**
- 跨域的网络限制与差异较大，难以通过k8s单集群支持
- 当前单集群的规模为**1k到2k**（k8s社区1.3版本规模为2k，年度目标为5k）
- 根据目前的应用增长速度，容器化之后整体规模需要支持**3w虚机，10w容器**
  1）按虚机应用算年底VM数量将超过6万
  2）应用/实例的部署比例平均在1:2到1:3之间
  3）容器化1个应用平均拆分出4~5个微服务

多集群联邦架构 复用list-watch机制实现组件解耦



Ubernetes Api server

增加集群相关API，屏蔽集群差异，统一请求入口

Ubernetes Scheduler

分拆联邦级别对象到集群

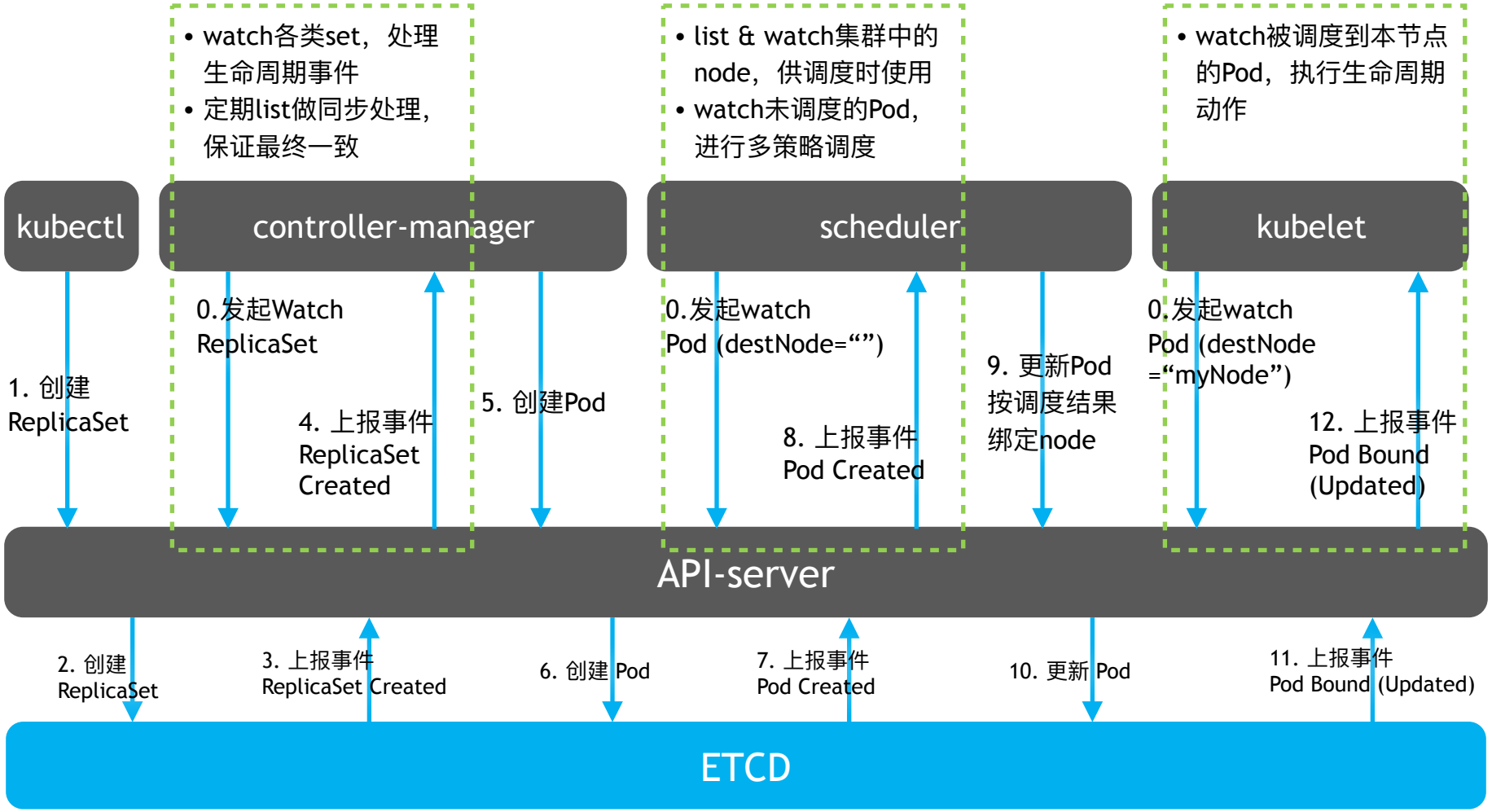
Cluster controller

管理个集群状态，集群级别对象创建

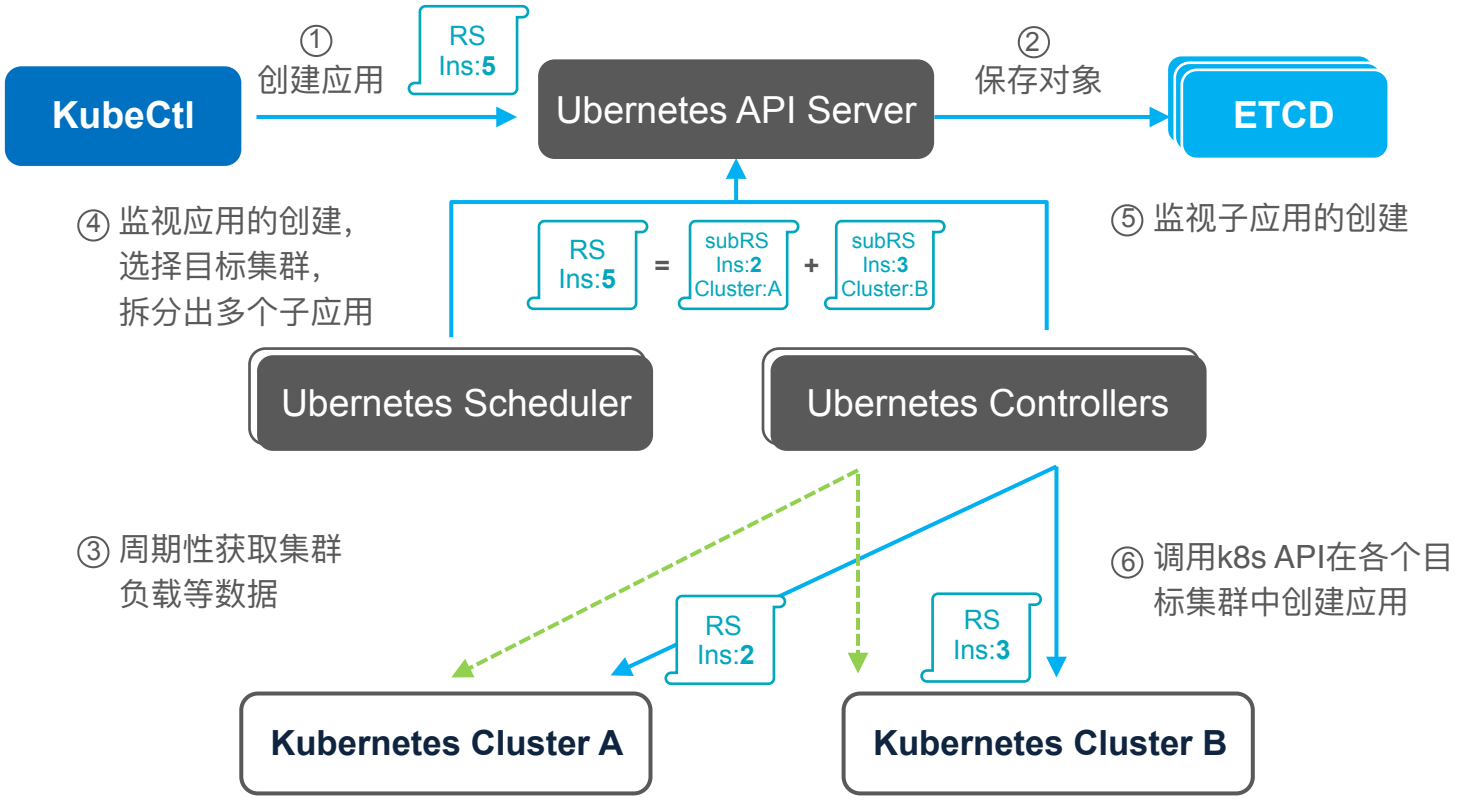
Service controller

跨集群服务发现

Kubernetes的list-watch机制



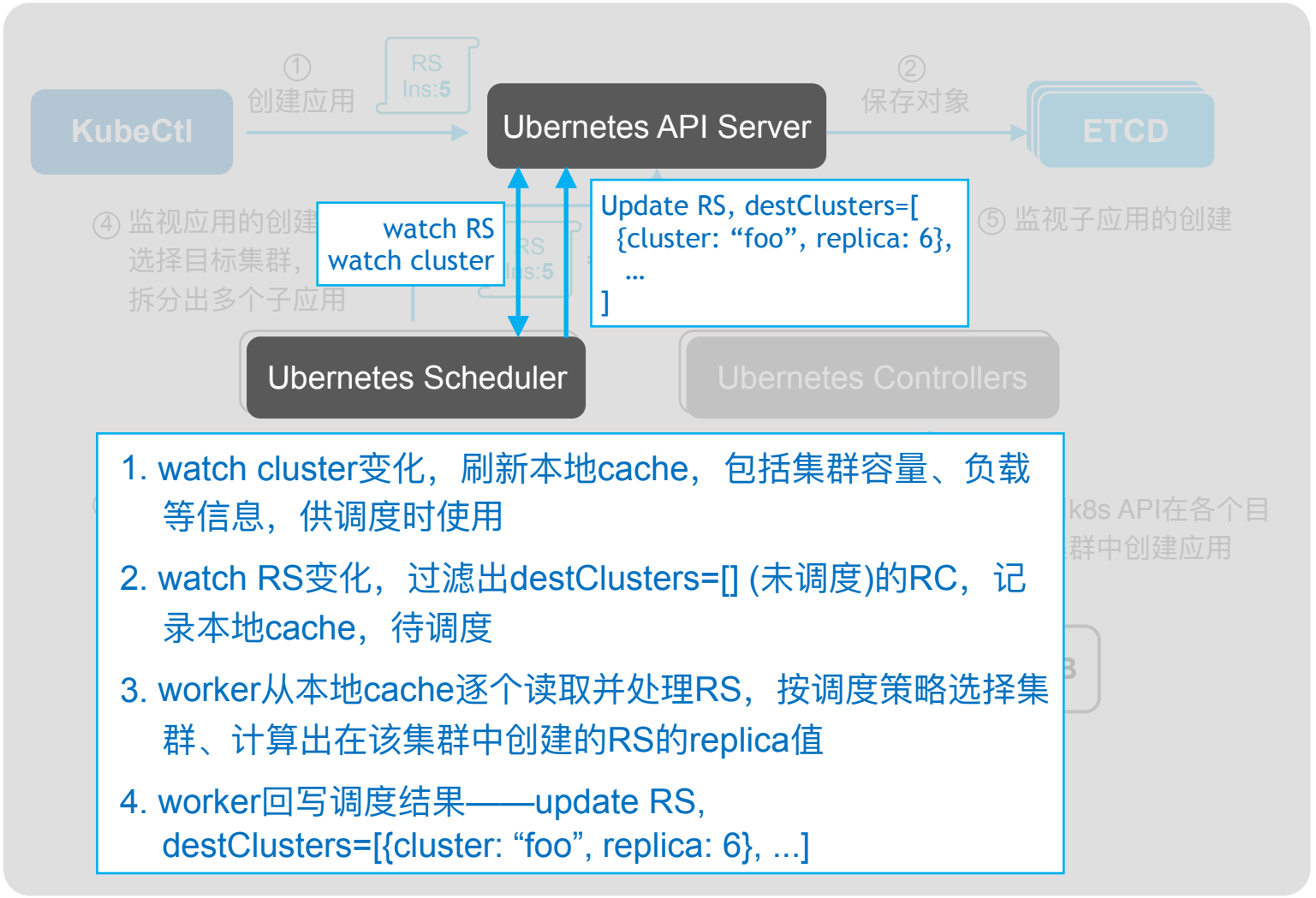
多集群联邦下的应用创建



RS: ReplicaSet多实例无状态应用

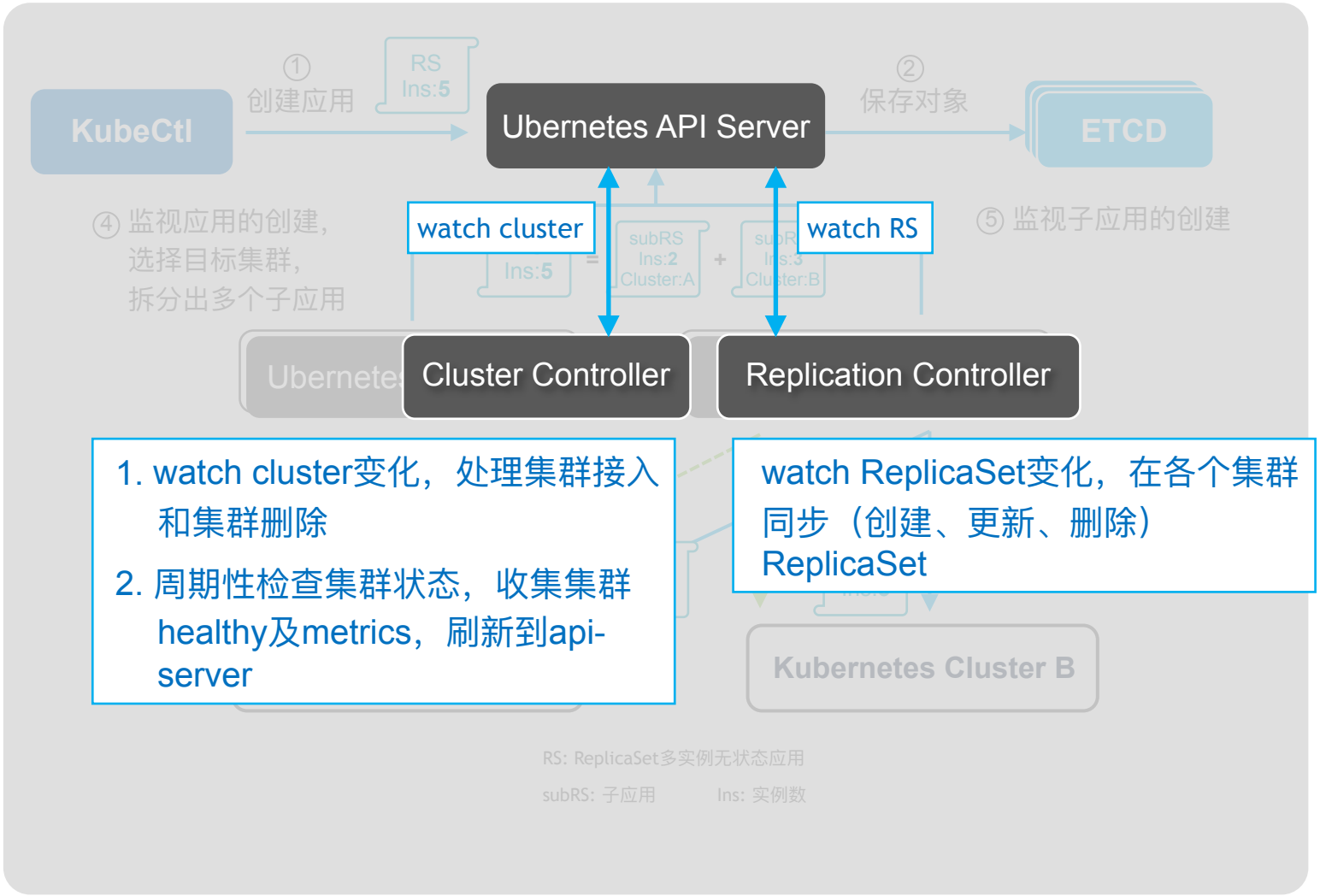
subRS: 子应用　　Ins: 实例数

联邦调度器 分拆联邦级别对象到集群

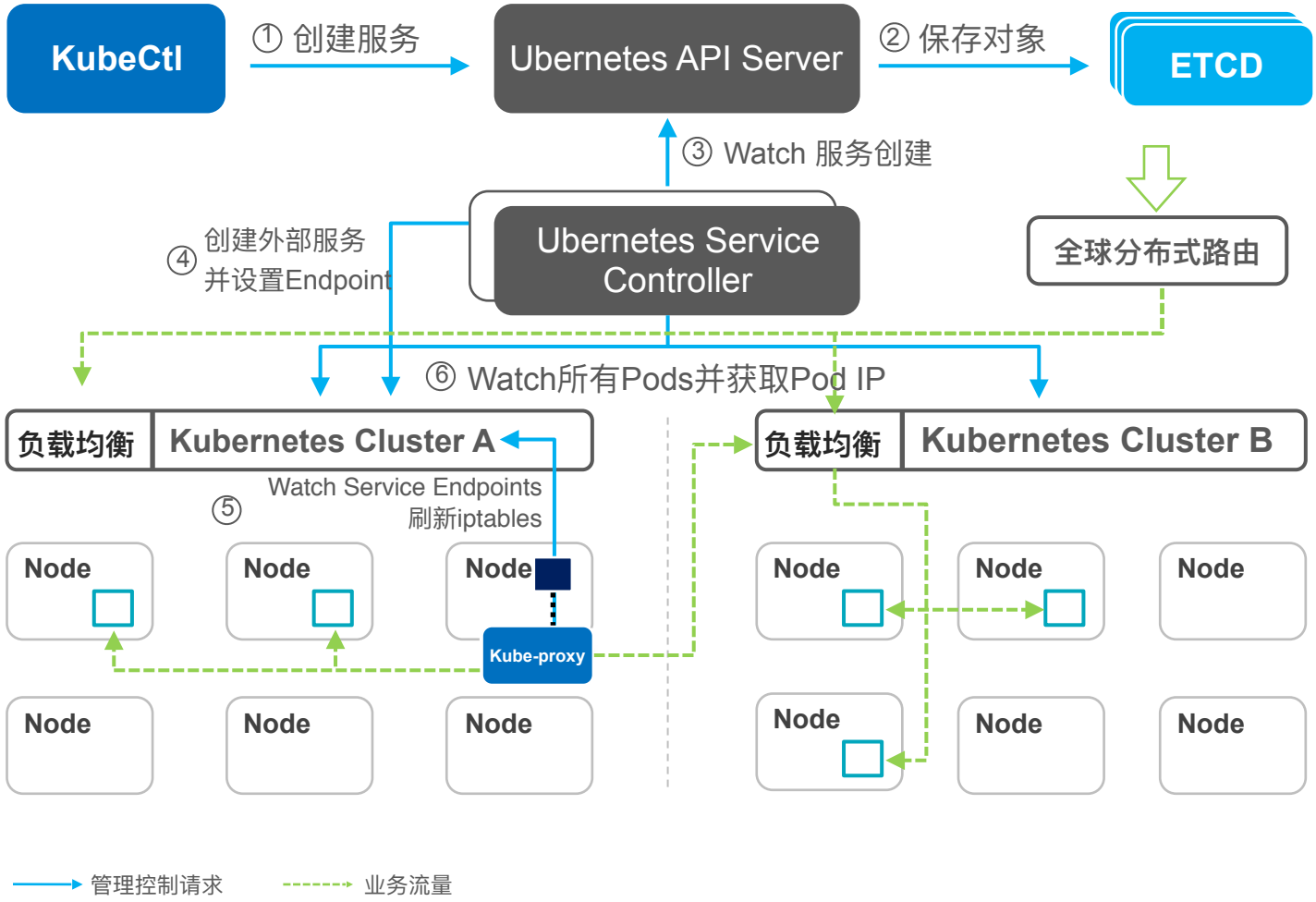


1. watch cluster变化，刷新本地cache，包括集群容量、负载等信息，供调度时使用
2. watch RS变化，过滤出destClusters=[] (未调度)的RC，记录本地cache，待调度
3. worker从本地cache逐个读取并处理RS，按调度策略选择集群、计算出在该集群中创建的RS的replica值
4. worker回写调度结果——update RS, destClusters=[{cluster: “foo”, replica: 6}, ...]

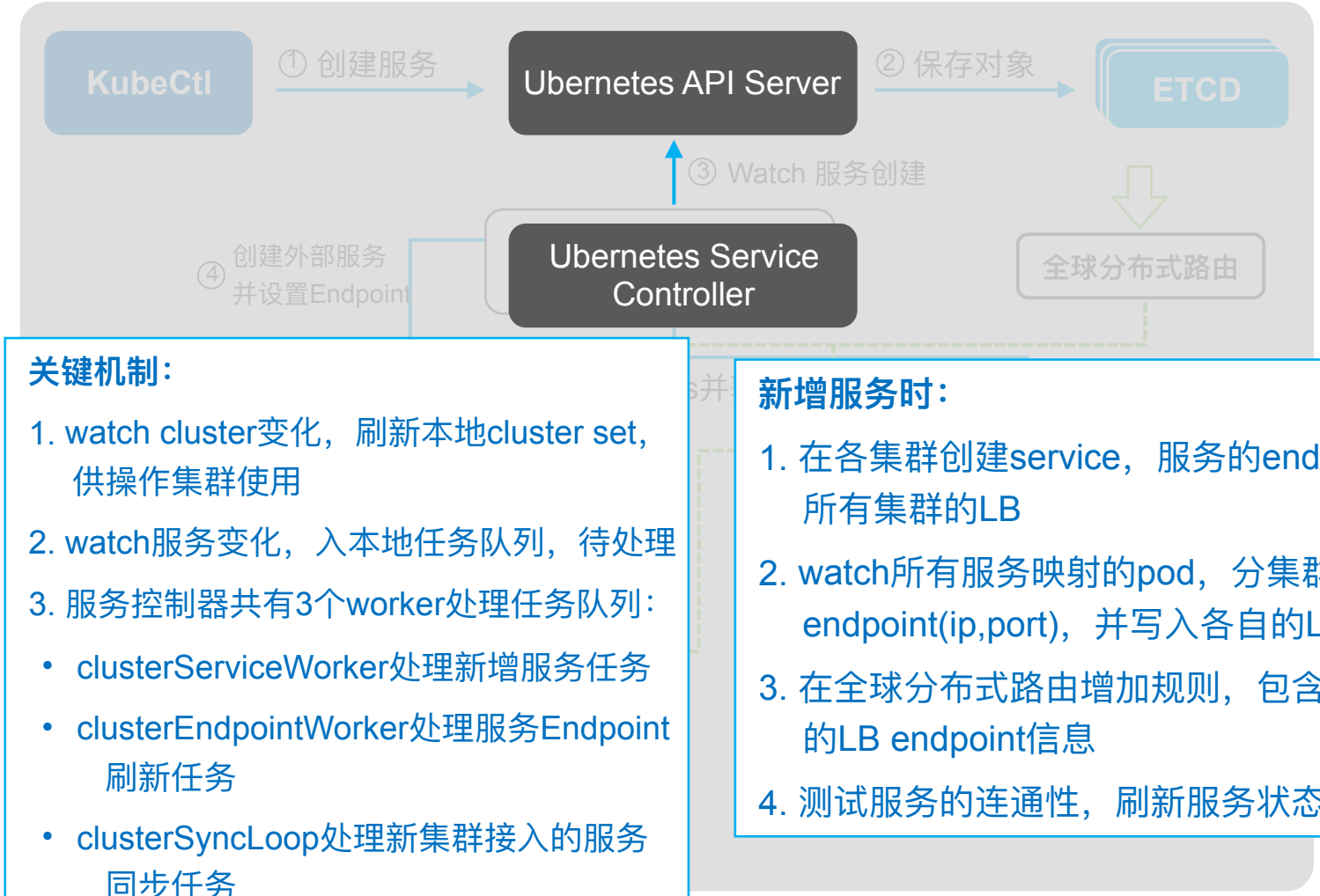
联邦集群控制器、副本集控制器



多集群联邦下的服务发现



联邦服务控制器跨集群服务发现



关键机制：

1. watch cluster变化，刷新本地cluster set，供操作集群使用
2. watch服务变化，入本地任务队列，待处理
3. 服务控制器共有3个worker处理任务队列：
   - clusterServiceWorker处理新增服务任务
   - clusterEndpointWorker处理服务Endpoint刷新任务
   - clusterSyncLoop处理新集群接入的服务同步任务

新增服务时：

1. 在各集群创建service，服务的endpoint包含所有集群的LB
2. watch所有服务映射的pod，分集群收集endpoint(ip,port)，并写入各自的LB
3. 在全球分布式路由增加规则，包含所有服务的LB endpoint信息
4. 测试服务的连通性，刷新服务状态

应用间的亲和/反亲和调度

A 亲和性

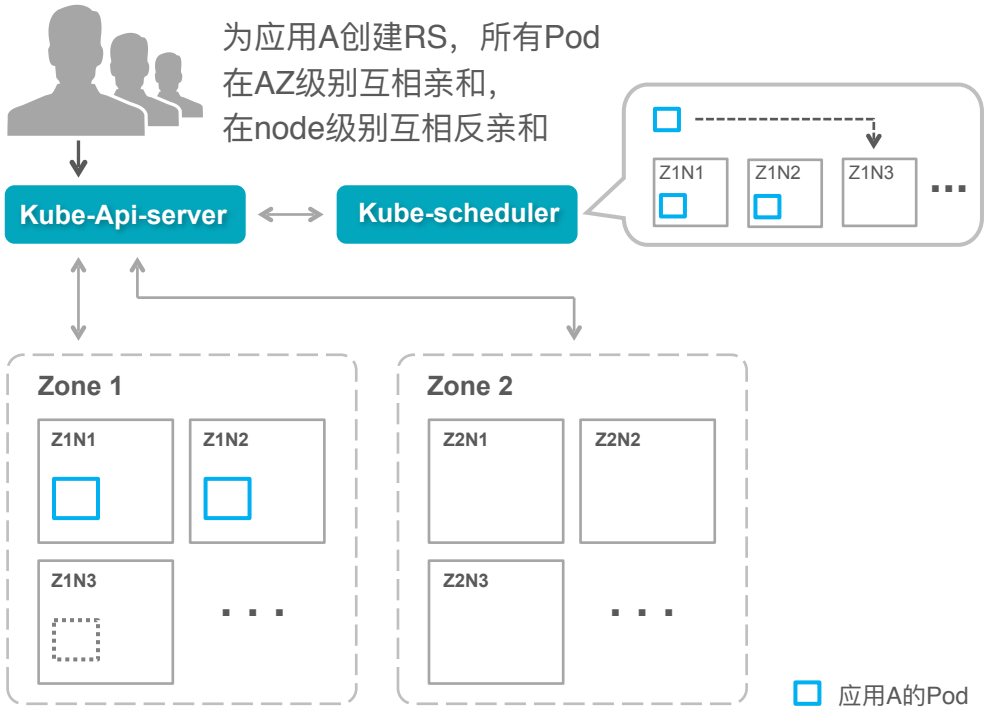
传统应用进行容器化，拆分微服务之后的部署约束，需要按实例逐一配对**就近部署**，容器间通信**就近路由**，减少网络消耗。

B 反亲和性

高可靠性考虑，同个应用的多个实例反亲和部署，**减少宕机影响**

互相干扰的应用反亲和部署，避免干扰。

应用间的亲和/反亲和调度

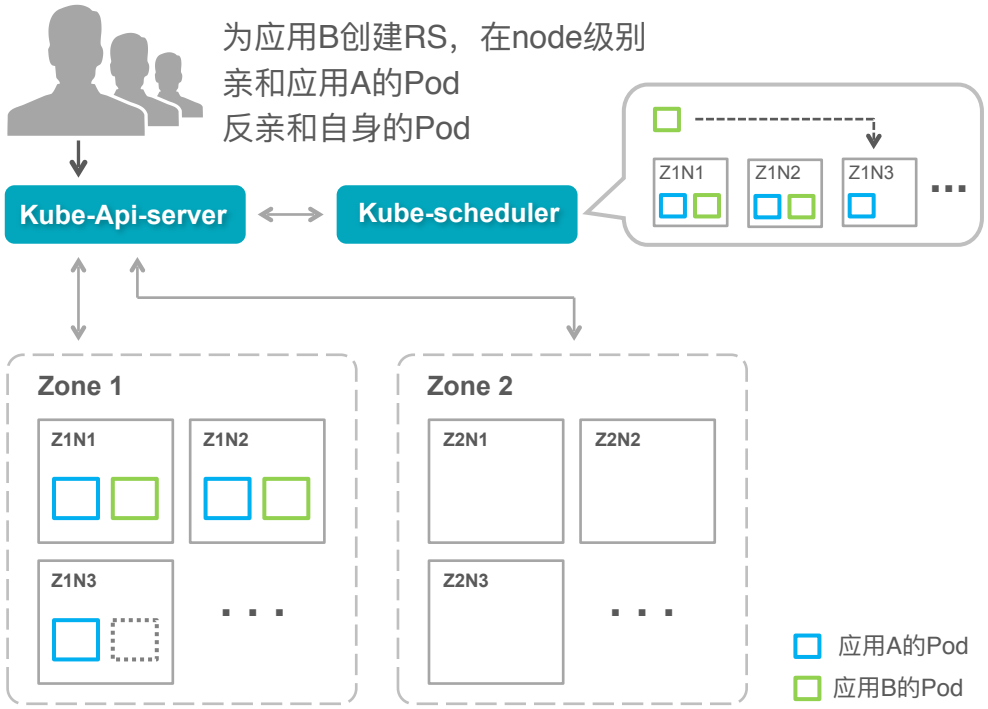


支持不同级别的亲和性

根据node按labels键值对“动态分组”

在这个例子中：计算亲和性时，所有az=zone1的node为一组，az=zone2的node为另一组。
计算反亲和性时，所有node各为独立的组。

应用间的亲和/反亲和调度



支持硬性亲和/反亲和

调度时**过滤**，符合的node保留，不符合的node直接排除

支持软性亲和/反亲和

调度时**评分**，符合的node高分，不符合的node低分

鸡生蛋蛋生鸡 – 亲和性调度的对称性考虑

应用B亲和A，应用A后于B创建怎么办?

对称性设计，调度B时会检查B是否被别人亲和/反亲和

被亲和的应用跑了（比如挂了异地重建）怎么办?

Pod挂掉通常kubelet会直接重新拉起。只有node挂掉时，node上所有的pod被终结，RC/RS重新创建Pod，才会有跑掉的情况。

鸡生蛋蛋生鸡 – 亲和性调度的对称性考虑

应用间亲和性（pod affinity）的对称性 – 不完全对称

检查待部署的pod亲和哪些已存在pod，同时检查该pod被哪些已存在pod亲和

hard pod affinity：不对称

正向的亲和为hard

反向的亲和为soft（不能因为没有被其他pod硬亲和就阻塞调度）

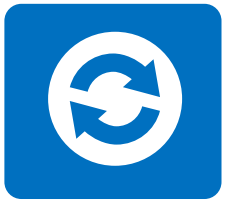
soft pod affinity：对称

正向反向都为soft

应用间反亲和性（pod anti-affinity）的对称性 – 完全对称

检查待部署的pod反亲和哪些已存在pod，同时检查该pod被哪些已存在pod反亲和。

对称性之后 – 未来可用的措施



限制异地重启（forgiveness）

Node挂掉时，pod不被终结，等待node恢复时由kubelet原地重启pod

运行时迁移（rescheduling）

Rescheduler周期性检查集群中pod的亲和/反亲和性规则，并迁移调整

后续考虑投入方向

规模 & 性能

前期的规模性能分析主要在控制面，后面将投入数据面分析。

网络规模和性能优化，以支撑10万容器的需求。

调度与重调度

Dedicated Node
Forgiveness(限制异地重启)
Rescheduling(运行时重调度)

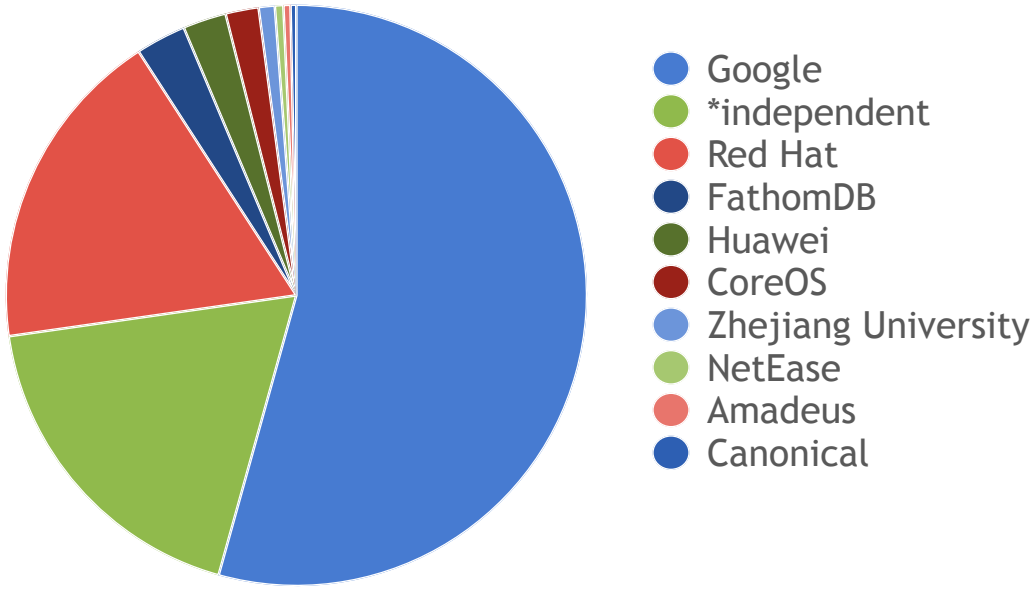
集群联邦

当前社区仅贡献了phase 1，尚不完整，需要持续投入。

Kubernetes开源社区贡献

commits 400+，全球排名第四，国内第一

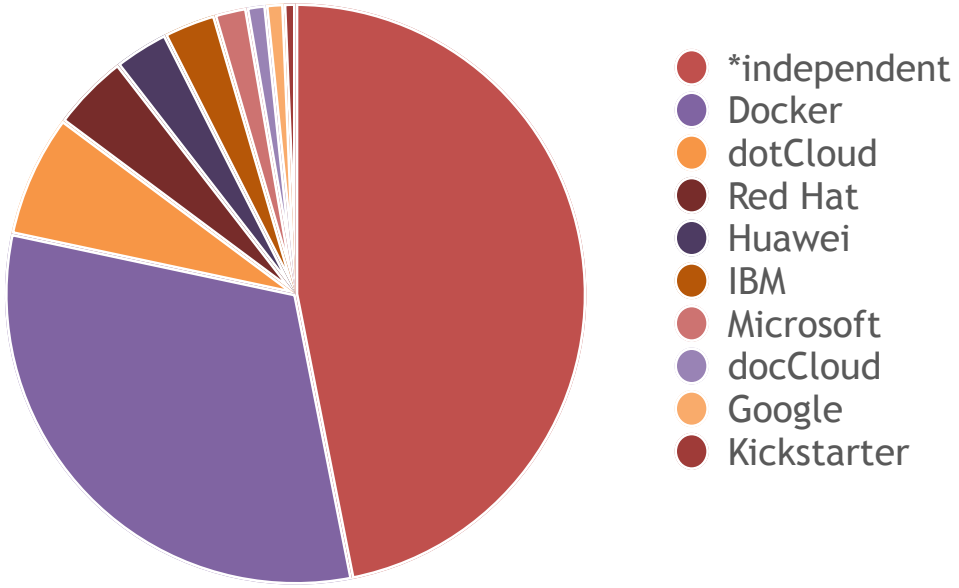
- CentOS k8s集群管理
- k8s运维系统对接kafka
- k8s运维系统对接elastic search
- Heapster重构及sink扩展
- 主导设计Ubernetes集群联邦
- Node Affinity
- Pod Affinity/Anti-Affinity
- Taints-tolerations
- 其他Bug和修复……



Docker开源社区贡献

commits 800+，全球排名第四，国内第一

- 增加关闭容器oom功能
- 容器重启策略增强
- 日志格式优化，更便于解析
- 环境依赖检测功能增强
- 增加容器cpu带宽限制
- 增加容器IO带宽限制
- 增加内存节点限制
- 增加内核内存限制
- 增加内存预留机制
- 增加swap内存限制
- Docker exec增强，增加指定用户和特权用户执行exec
- Docker build资源限制增强
- 增加ARM64支持
- 安全加固，增加seccomp支持
- 增加在容器内获取cgroup信息功能



云容器引擎即将公测 — 邀您体验

平台共建 · 市场合作 · 产品咨询

Thanks!