Serverless autoscaling in kubernetes

莫源
阿里云容器服务技术专家

个人简介

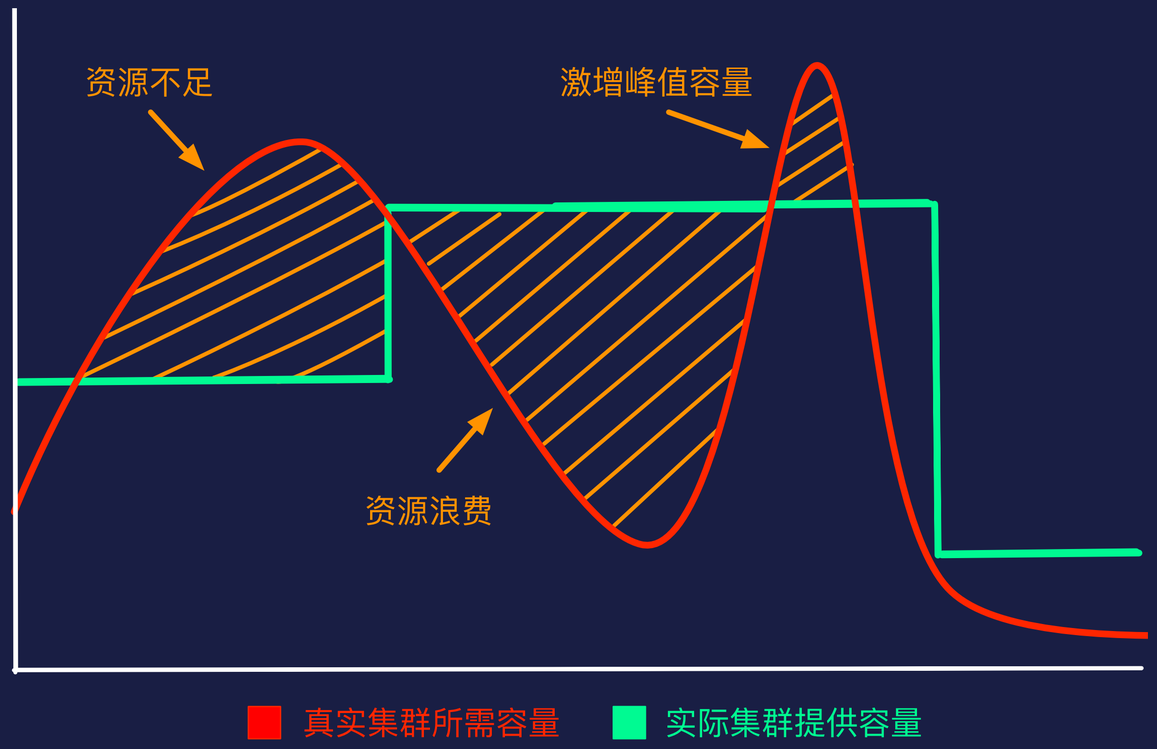
刘中巍，花名：莫源

2013年加入阿里云

先后负责阿里云控制台架构、容器服务管控系统、服务发现系统、持续集成与持续交付等方面的研发。

目前主要工作在容器可观测性、弹性伸缩与成本优化。kubernetes社区弹性相关组件的maintainer/reviewer。

当我们在谈论“弹性伸缩”的时候



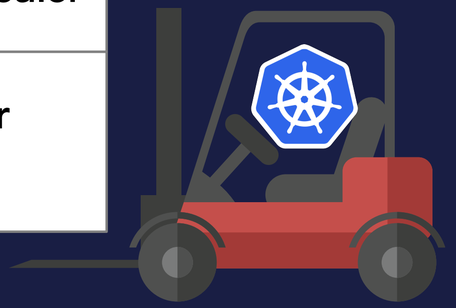
弹性伸缩对于不同角色的人员，有着不同的意义。

开发人员希望通过弹性伸缩使应用获得高可用的保障
运维人员希望通过弹性伸缩降低基础设施的管理成本
架构师希望通过弹性伸缩得到灵活弹性的架构应对突发的激增峰值。

弹性伸缩有多种不同的组件和方案，选择适合自己业务需求的方案是落地执行前的第一步。

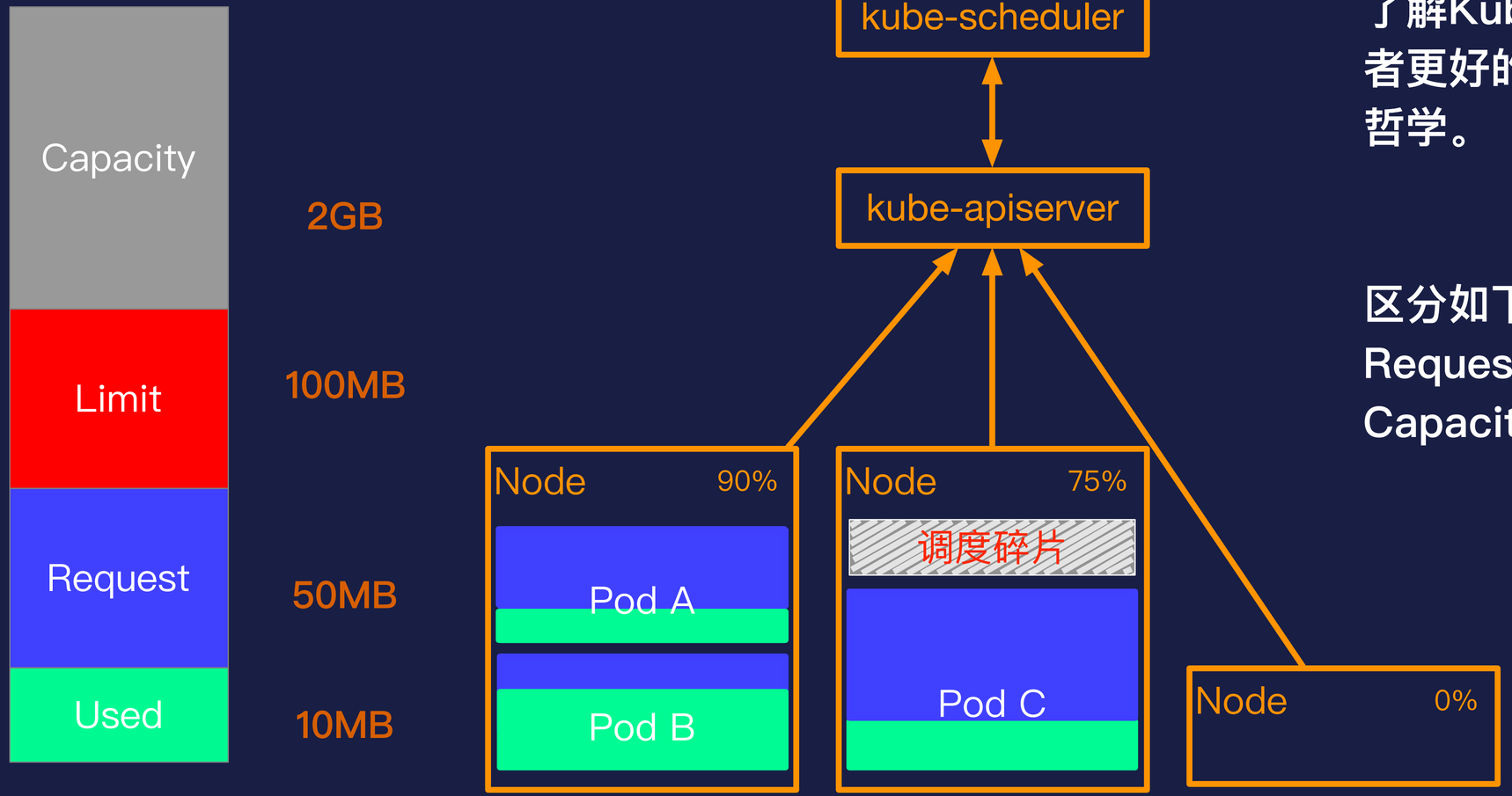
Kubernetes弹性伸缩的组件布局

	Nodes	Pods
Horizontal	cluster autoscaler	HPA cluster proportional autoscaler
Vertical	none	vertical pod autoscaler addon resizer



Kubernetes弹性伸缩组件可以从伸缩方向和伸缩对象两个维度进行解读，其中HPA与Cluster-Autoscaler是开发者最常组合使用的弹性伸缩组件。HPA负责容器的水平伸缩，Cluster-Autoscaler负责节点的水平伸缩。

Kubernetes弹性伸缩的挑战



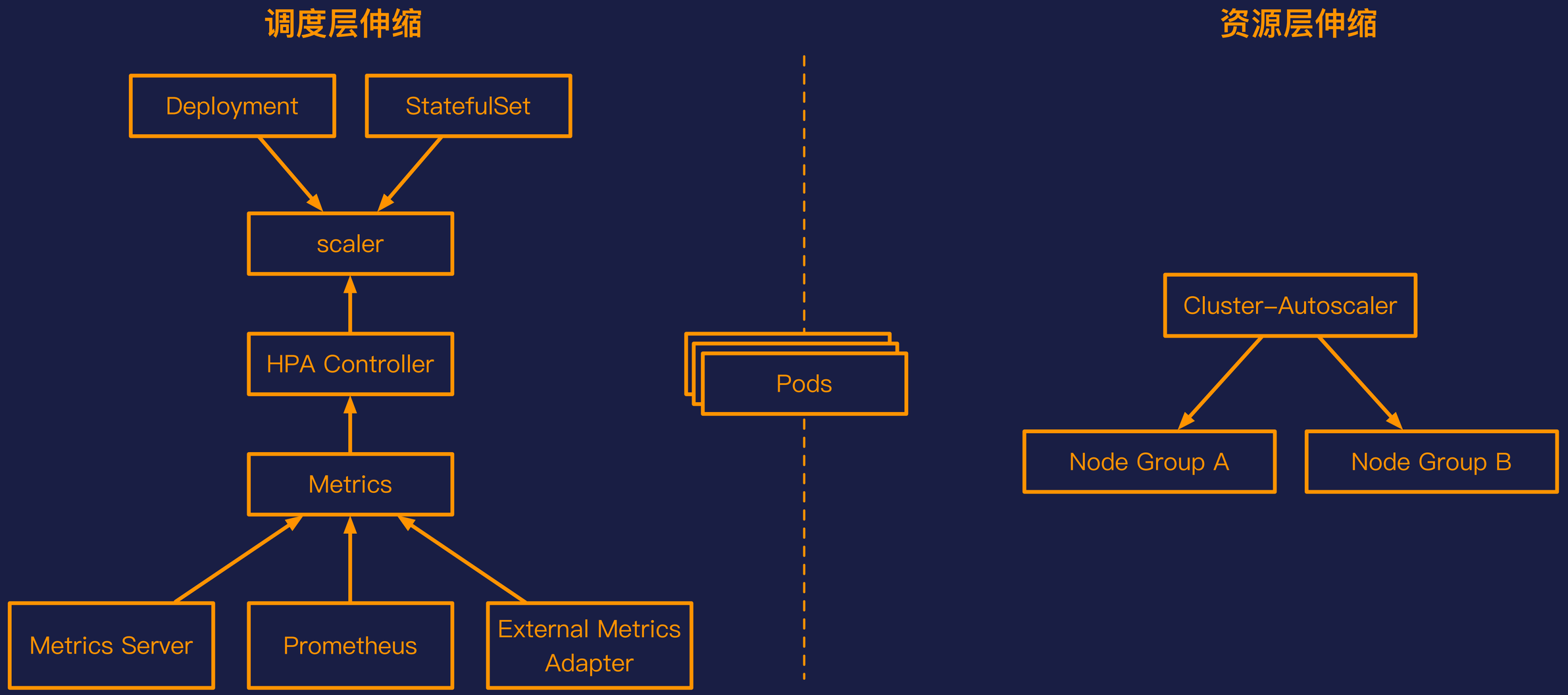
了解Kubernetes的调度方式可以帮助开发者更好的理解Kubernetes弹性伸缩的设计哲学。

区分如下四个概念：真实利用率、Request(调度）、Limit(存活）、Capacity(容量)

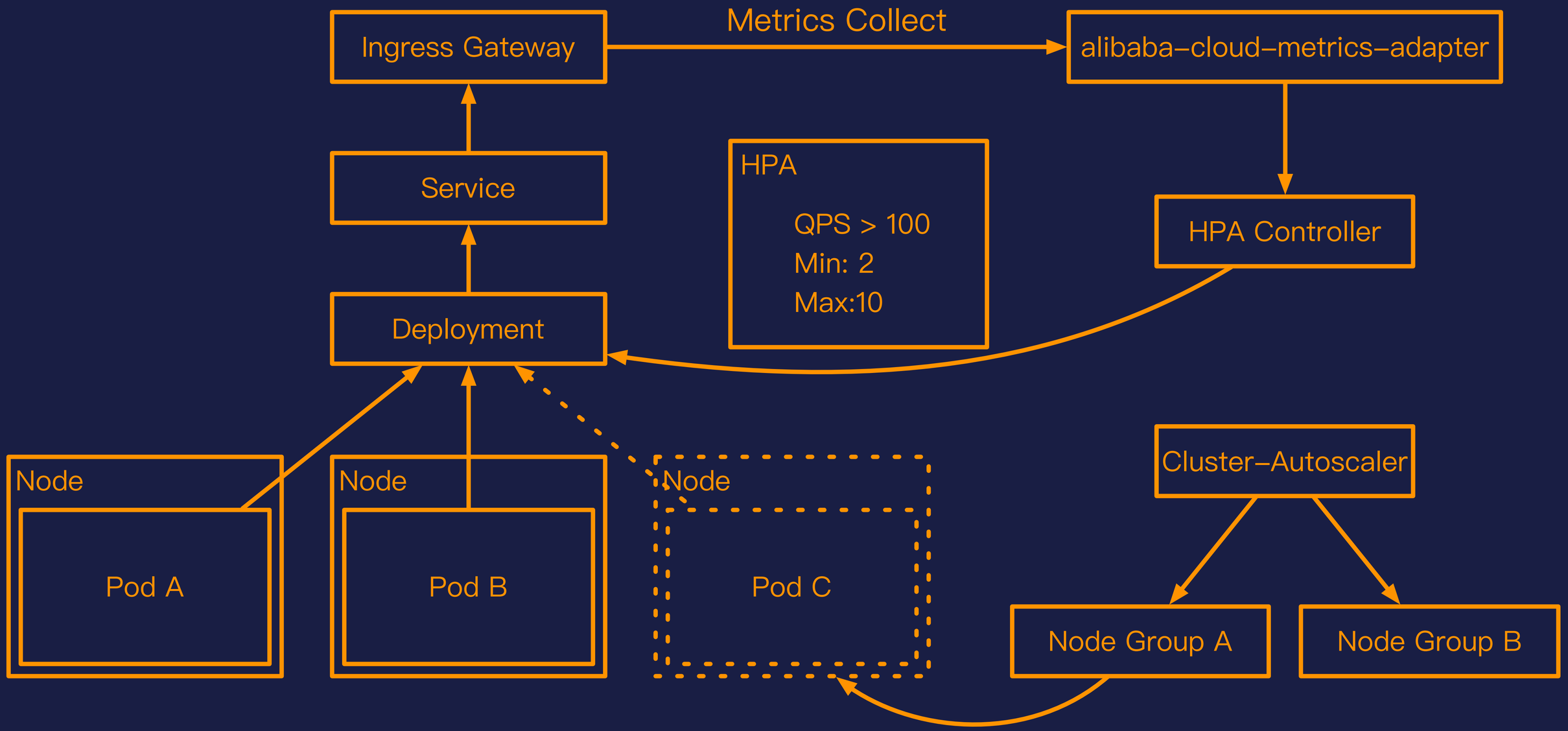
Kubernetes弹性伸缩的三大难题：

1. 容量规划炸弹
2. 百分比碎片陷阱
3. 资源利用率困境

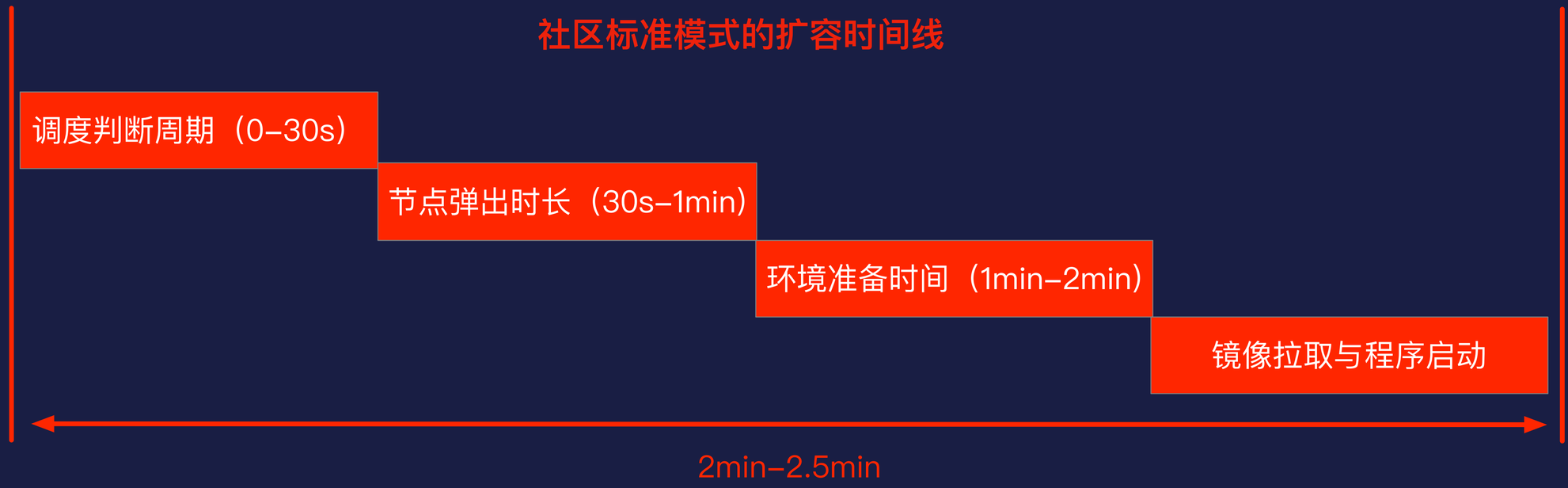
Kubernetes弹性伸缩的设计哲学

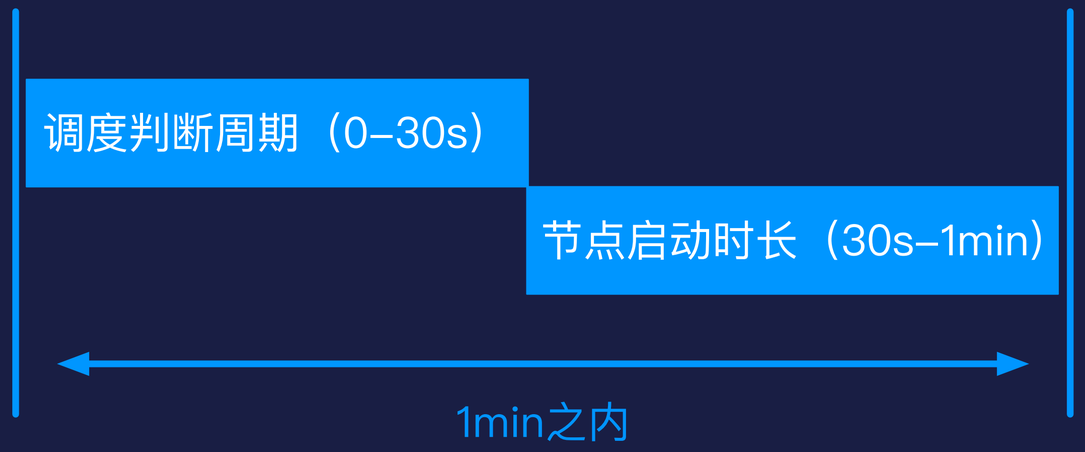


经典的Kubernetes弹性伸缩案例



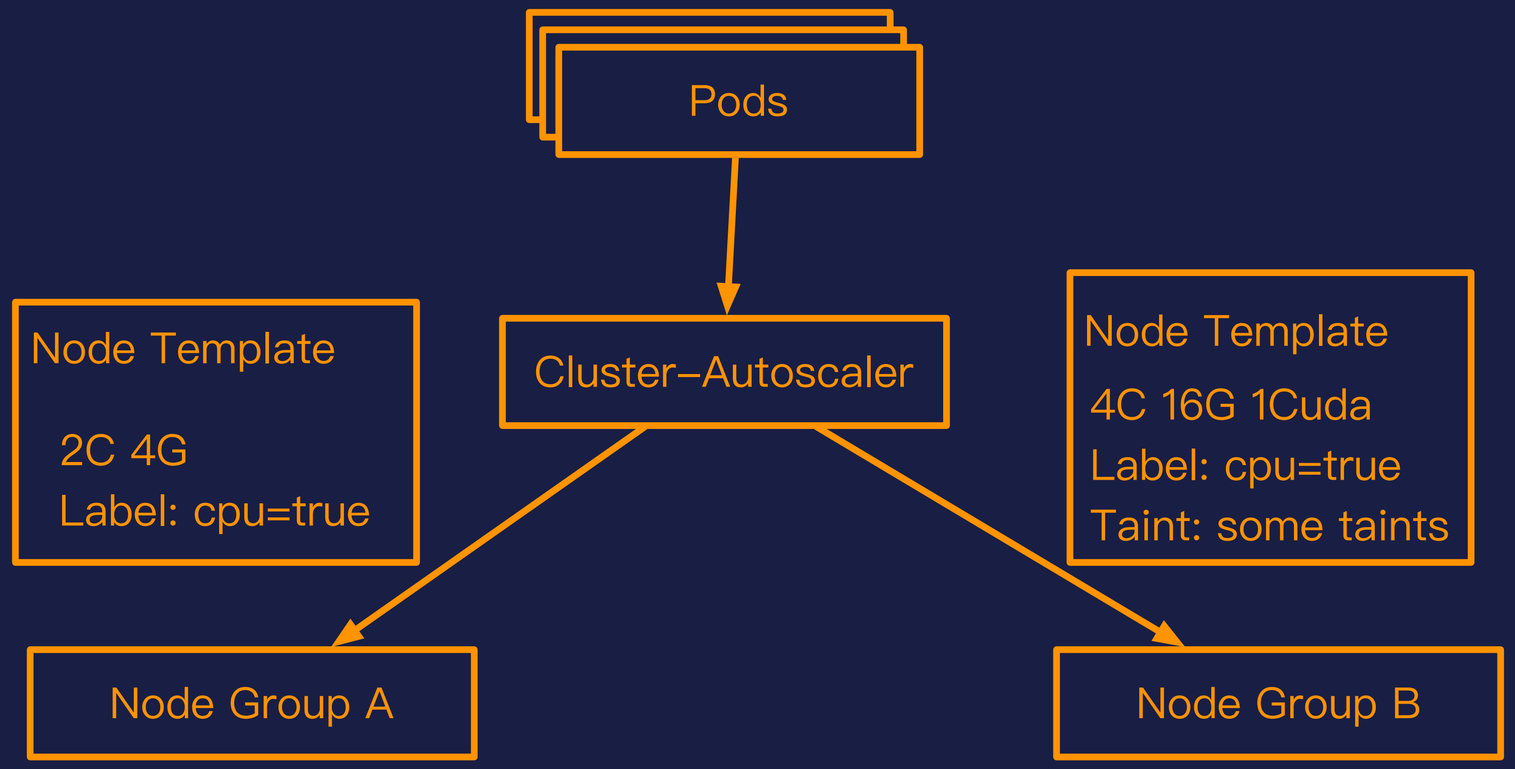
Kubernetes弹性伸缩的阿克琉斯之踵（1） – 时延





标准模式是创建、释放ECS的方式，扩容的时延在2min–2.5min左右，而阿里云独立的极速模式是通过创建、停机、启动的方式进行实现，停机时只收取存储的费用，不收取计算的费用。可以通过非常低廉的价格获得50%以上的弹性效率

Kubernetes弹性伸缩的阿克琉斯之踵（2）– 复杂度



扩容时为了保障伸缩出的机器可以满足调度条件，cluster–autoscaler会使用node–template的机制进行模拟调度，只有模拟调度能够满足条件的node group才可以触发节点的添加。

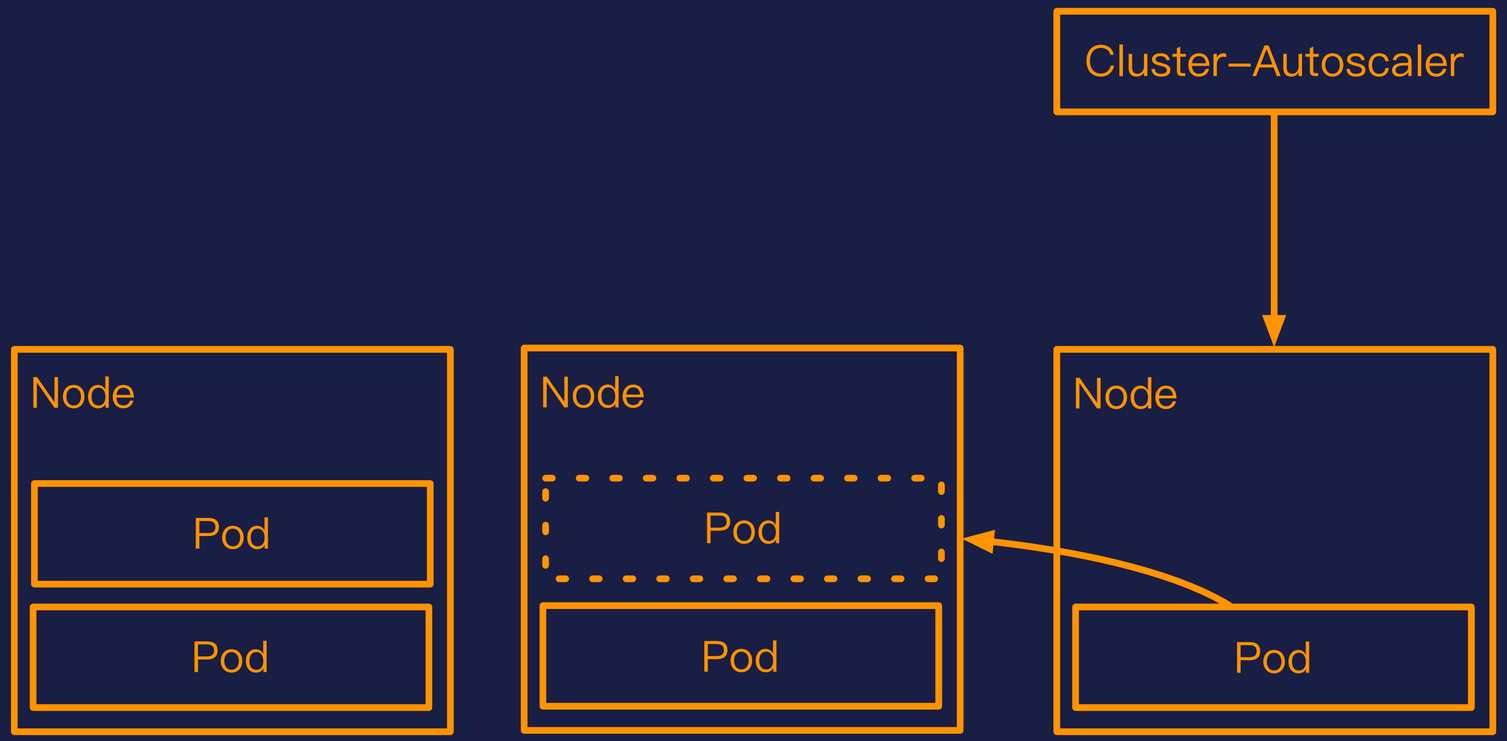
可能扩容失败的原因：

1. 机型规格配置不匹配
2. 调度信息不匹配
3. 伸缩组状态不符合条件
4. 库存等原因造成失败等。

解决方案：

1. 多伸缩组多规格解决配置的问题

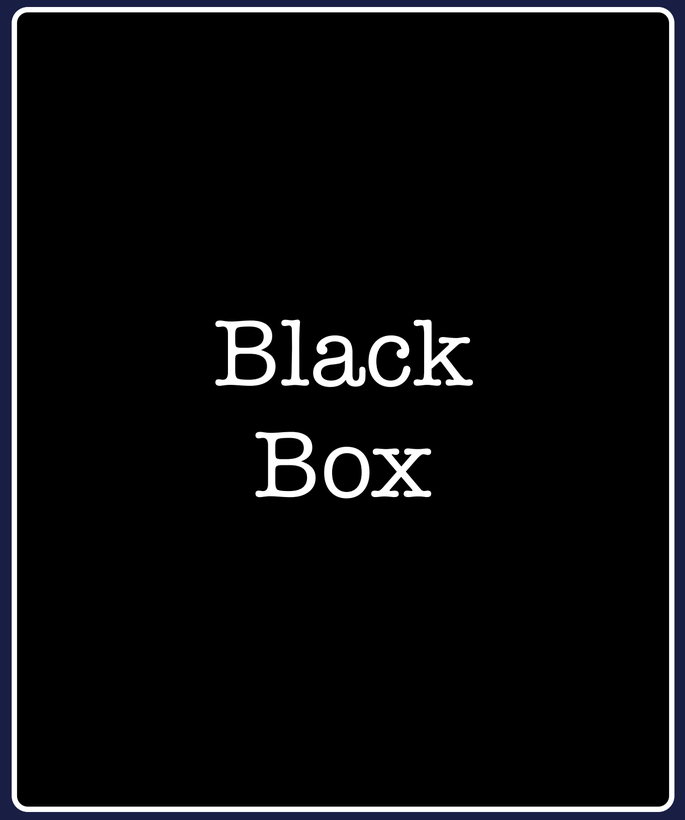
Kubernetes弹性伸缩的阿克琉斯之踵（2） – 复杂度



缩容时需要考虑的问题是当节点上还残存Pod时该如何排水驱逐。特别是出现伸缩失败等异常场景时的恢复处理机制。

cluster–autoscaler内部逻辑通过模拟调度、预排水、状态机式的故障恢复策略进行逻辑判断的增强与恢复。

Kubernetes弹性伸缩的阿克琉斯之踵（3）–学习成本

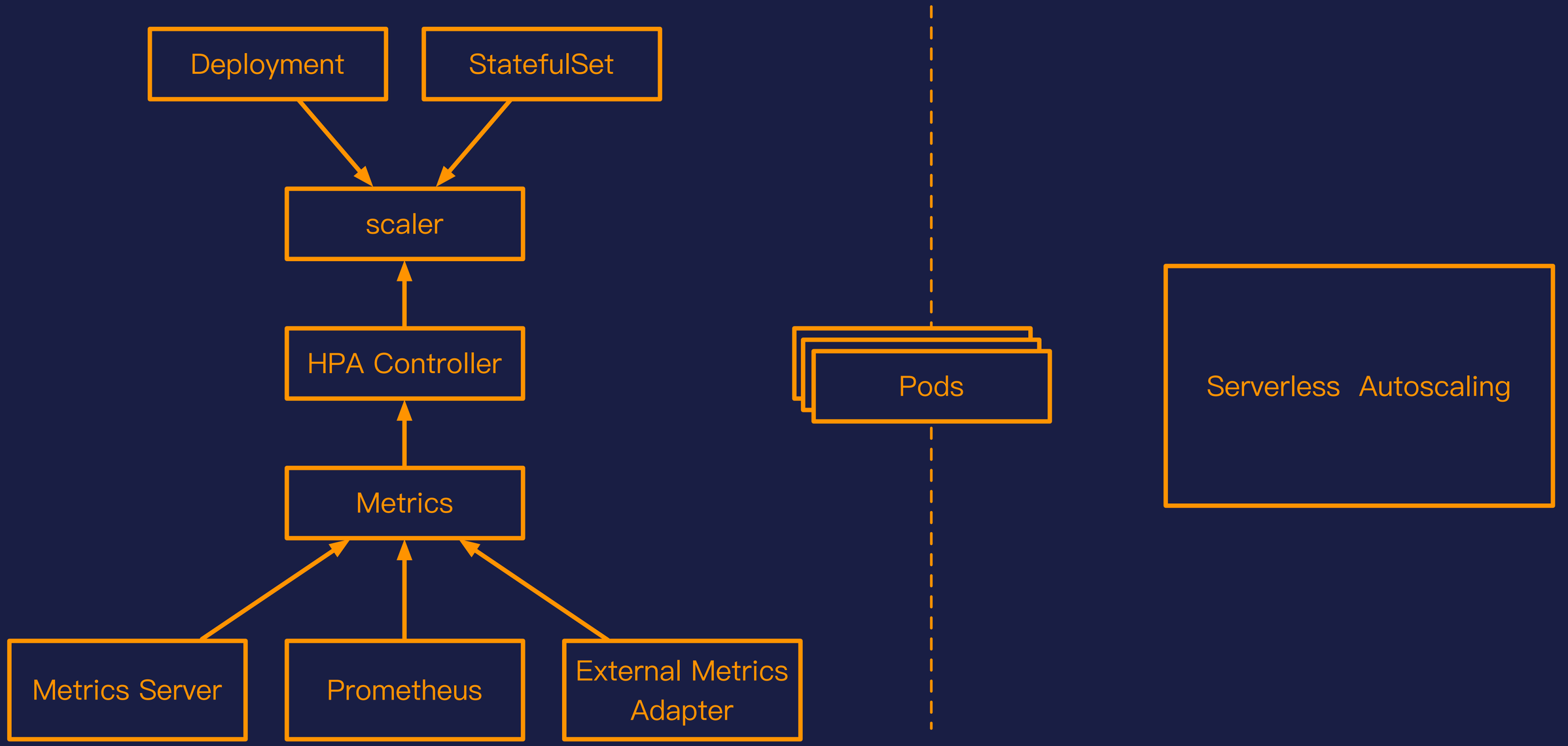


对于大多数的开发者而言，cluster-autoscaler的工作原理是黑盒的，而且cluster-autoscaler目前最好的问题排查方式依然是查看日志。

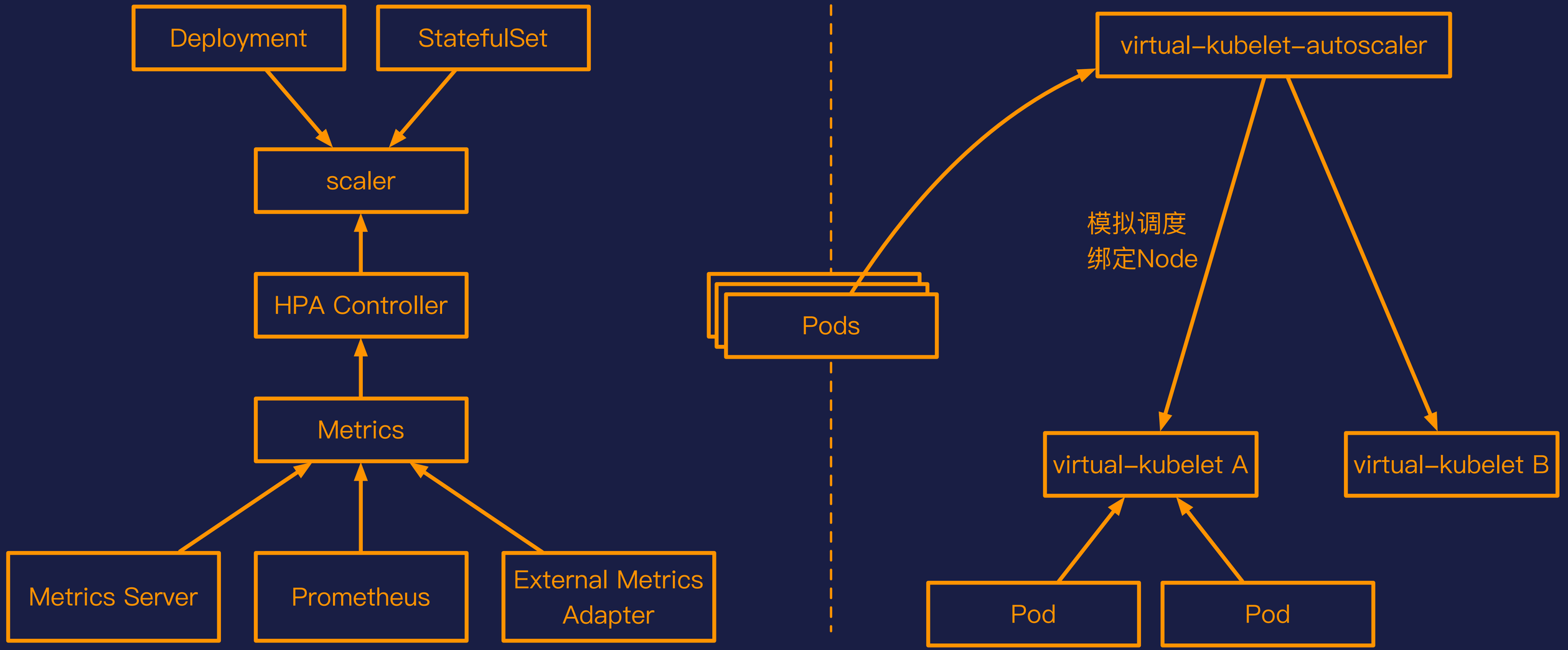
一旦cluster-autoscaler出现运行异常后者由于开发者配置错误导致无法如预期的伸缩，那么80%以上的开发者是很难自己进行纠错的。

阿里云容器服务团队开发了一款kubectl plugin，可以提供cluster-autoscaler更深层次的可观测性，可以查看当前cluster-autoscaler所在的伸缩阶段以及自动弹性伸缩纠错等能力。

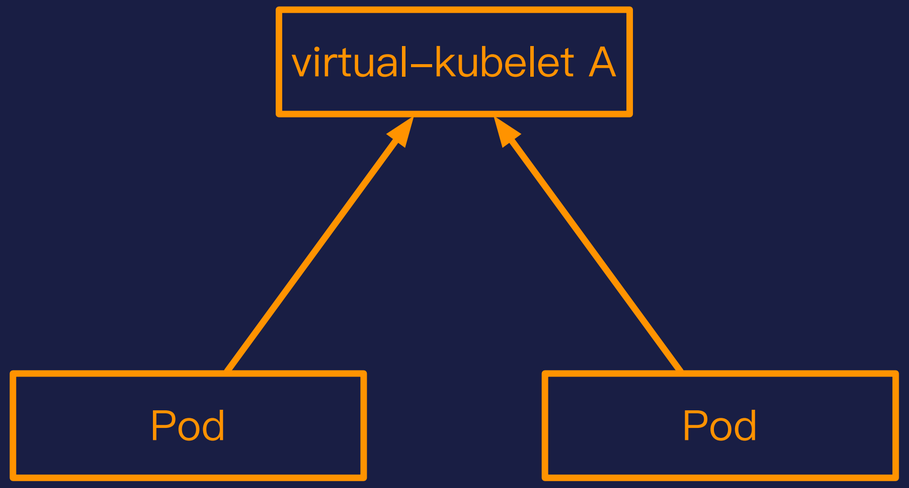
阿克琉斯的马丁靴 – Serverless autoscaling



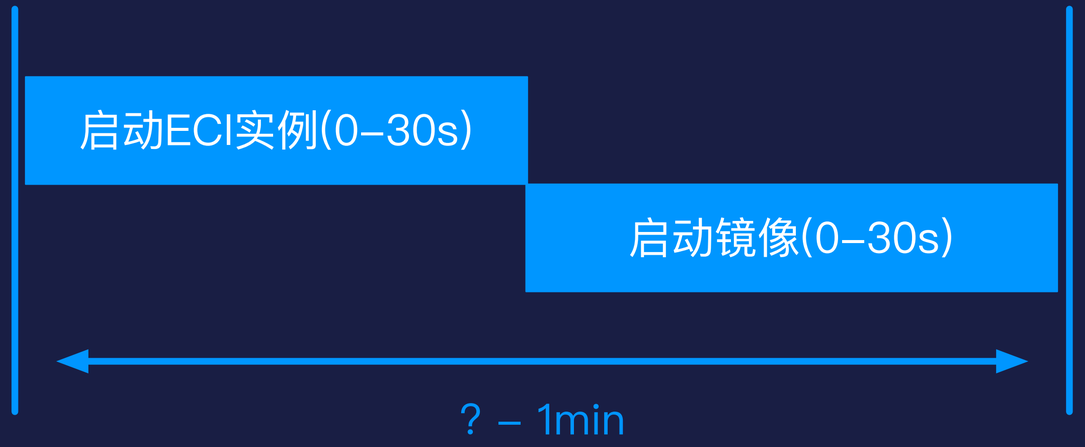
Serverless autoscaling组件 – virtual-kubelet-autoscaler



virtual-kubelet-autoscaler原理解析（1）



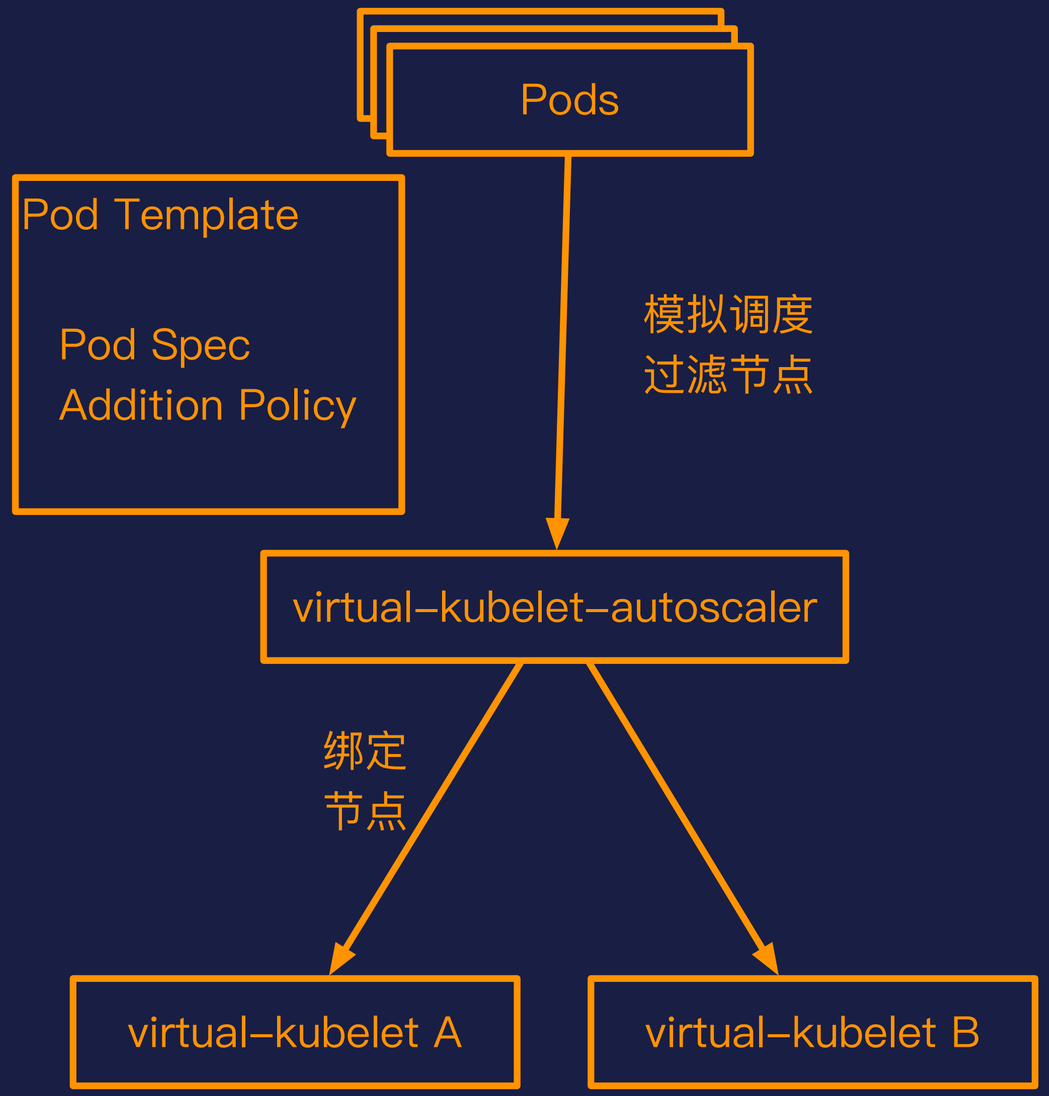
virtual-kubelet负责承载真实的负载，可以理解为一个虚拟节点，拥有无限大的capacity。当Pod调度到virtual-kubelet上时，会将Pod通过轻量级实例ECI进行启动。



目前ECI的启动时间在30s之内，程序从调度开始到运行一般会在1分钟内拉起。

ECI新架构支持极速启动，最短可缩短到？秒

virtual-kubelet-autoscaler原理解析（2）



与cluster-autoscaler类似，virtual-kubelet-autoscaler也需要使用模拟调度的机制来判断Pod是否可以被真实处理和承载，但是相比cluster-autoscaler而言，存在如下差异。

1. virtual-kubelet-autoscaler模拟调度的对象是增加了调度策略的Pod Template并非Node Template。

2. virtual-kubelet-autoscaler的核心是选择virtual-kubelet来承载负载，一旦Pod模拟调度成功绑定到virtual-kubelet上后，Pod的生命周期管理、问题的排查等就与传统的Pod没有差异，不再是黑盒排查问题。

virtual-kubelet-autoscaler不是银弹

virtual-kubelet-autoscaler并不是用来替代cluster-autoscaler的，virtual-kubelet-autoscaler的优势在于使用简单、高弹性高并发，按量按需计费。

但是与此同时也牺牲了部分的兼容性，目前对cluster-pi、coredns等机制支持的还并不完善，只需少许的配置virtual-kubelet-autoscaler是可以和cluster-autoscaler兼容的。

virtual-kubelet-autoscaler特别适合的场景是大数据离线任务、CI/CD作业、突发型在线负载等。

Thank you !

扫码关注公众号,获取 831 深圳站 PPT