Kubernetes的落地与生态

才云科技

Kubernetes预期解决的企业痛点

机器、业务、新技术、数据多，成本做火箭上窜

业务增长造成IT系统扩张；更多的机器带来硬件成本、维护成本；日新月异的技术蹿升学习升本和错误成本；数据多但无法发挥价值

用户、业务需求瞬息万变，如何快而不乱

企业希望其线上业务能够实时更新，开发速度变的更敏捷，然而现实的笨重和错误带来具大经济损失

信任、安全是先天不足，企业已有IT设施投入无法释怀

云厂商涉足多个业务形成潜在商业利益冲突；企业不希望将鸡蛋放在同一个篮子里；企业无法放弃已有的IT设施投入

企业迈向云化计算缺乏技术积累，反而提升成本加大风险

云化平台在带来好处的同时极大提升了复杂度和维护难度

Kubernetes预期解决的企业痛点

物理资源分散，缺乏动态调度、系统成本高

为不同应用手动分配各自的服务器，手动操作成本高、速度慢，物理资源利用率低

缺乏容错和自我修复能力

出现单点故障导致系统宕机、无法根据用户根据业务自定义的规则进行健康检查、自动修复

缺乏高可用和负载均衡

应用程序存在单点失效风险，缺乏有状态、无状态自动负载均衡

缺乏弹性伸缩机制

缺乏统一的控制平台和可视化界面，需要借助零散的脚本完成不同任务，难以维护

应用程序有状态、集群化，难以管理和维护

应用有状态，难以动态迁移、起停、伸缩；中间件(如Redis)集群化，部署、配置、维护复杂

Kubernetes预期解决的企业痛点

开发、测试、生产环境不一致

应用依赖关系复杂，开发、测试、生产环境不一致，无法无缝将应用切换于不同环境

交付流程复杂、新版本发布风险大

代码构建、发布需要繁琐配置或人工处理；出错后难以快速回滚，缺乏灰度发布或滚动更新

环境配置繁琐

手动维护组件的IP、应用配置、外部依赖(Oracle, Web Logic)，且配置随环境不同而变化

操作使用众多随机的脚本

缺乏统一的控制平台和可视化界面，需要借助零散的脚本完成不同任务，难以维护

系统调试困难、缺乏有效监测预警

出了问题难以快速定位，日志缺乏搜索、聚合等功能，实时监测工具、可视化、自动预警匮乏

Kubernetes预期解决的企业痛点

如何选择最合适的节点运行任务	可定制化的调度器会选择最适合的节点（VM或物理主机）来进行动态调度（可轮询、根据节点特性、或根据业务特性自定义规则）
组件或应用出现故障如何自动检测与修复	通过内置的节点层面、容器层面、应用层面的可根据业务定制的健康检查机制发现问题并发送预警和修复
如何应对不可测的流量变化、高并发	通过设置应用层面的自动伸缩策略来进行弹性扩容应对高并发，并在系统资源闲置时自动缩减
如何应对组件的IP变化、多实例负载均衡	面向服务，进行自动服务注册，服务具有固定IP，并提供DNS服务发现；支持服务多实例间的**L4**或**L7**负载均衡（有状态或无状态）功能

Kubernetes在生产中的功效

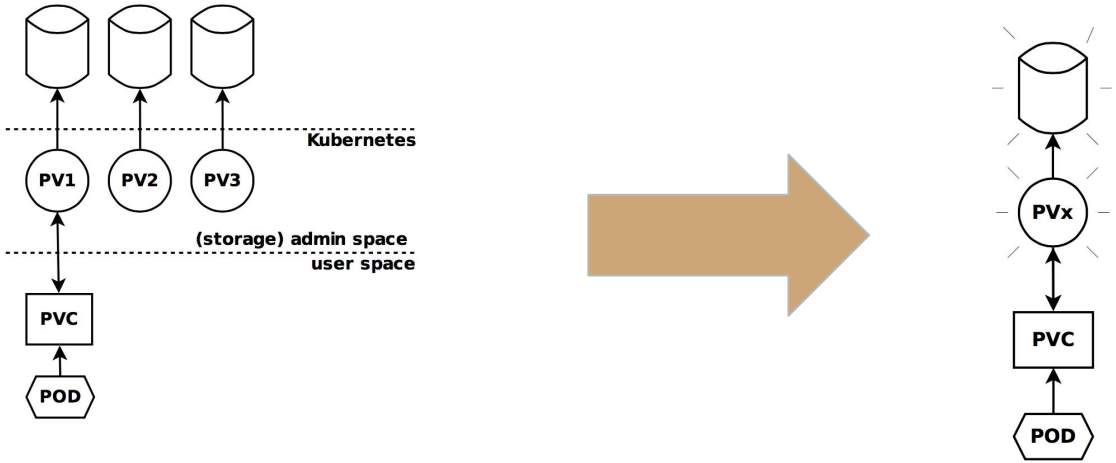
机器、应用多了以后如何根据逻辑关系管理	通过标签系统和查询接口来根据逻辑、组织结构梳理和管理系统内的资源和组件
微服务细粒度拆分后如何应对管理成本陡增	通过使用和定义容器组，将关系密切的模块统一管理统一调度减少跨模块调用
如何管理不同环境中的系统外部依赖和配置	内置配置服务提供统一的配置管理接口，应用可动态获取配置信息，完全实现应用与环境解耦合、应用与配置解耦合
对于有本地数据以来的应用如何做到迁移、扩容	可以使用节点亲和性调度，也可以使用**Persistent Storage Claim**，将底层的存储资源池化

Kubernetes在生产中的功效

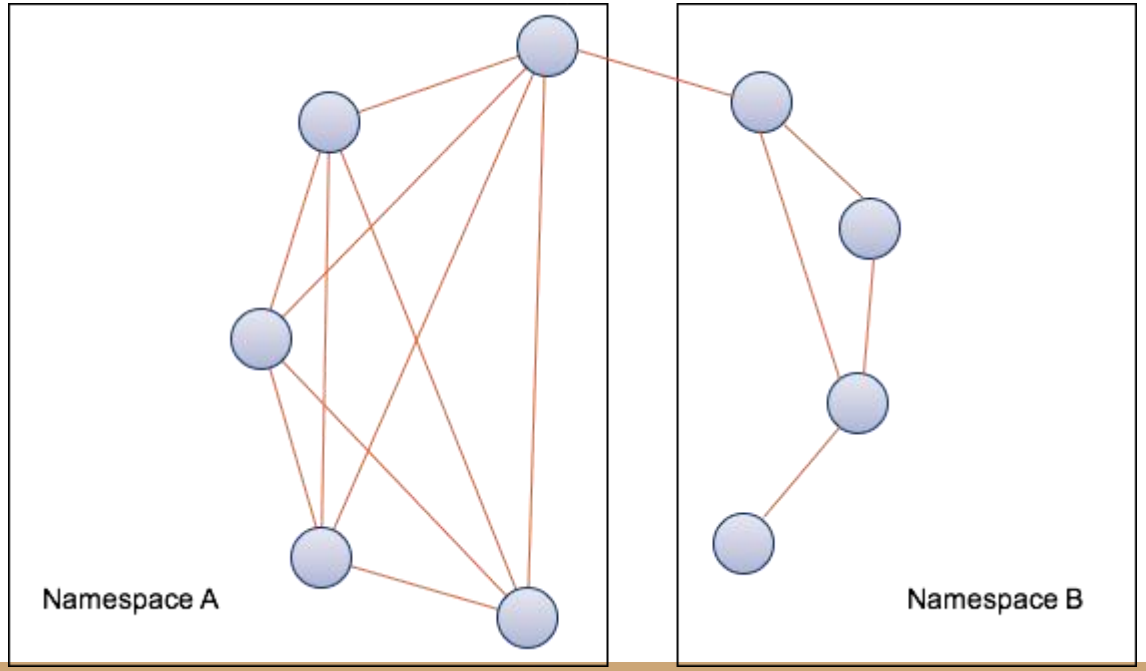
如何保证发布新版本时不掉线、零风险	通过滚动升级来保证新老版本可以无缝切换，而无需搭建传统的绿、蓝环境；通过灰度发布屏蔽新版本上线风险
如何支持有状态的“宠物型”应用	通过使用和定义节点亲和性调度来保证宠物应用运行在指定的特殊机器上; 或通过容器生命周期接口保证应用能正常的启动和停止
如何在物理集群上做细粒度资源划分和权限管理	Kubernetes内置namespace和service account提供集群资源的逻辑切分和隔离，提高系统安全性
如何支持批处理任务、守护进程任务	使用Kubernetes自带的Daemon Set和Jobs来轻松支持各种应用类型

更完善的存储管理

Dynamic Storage Provisioning: Storage class, External provisioning



更完善的网络管理



更方便的有状态传统应用管理

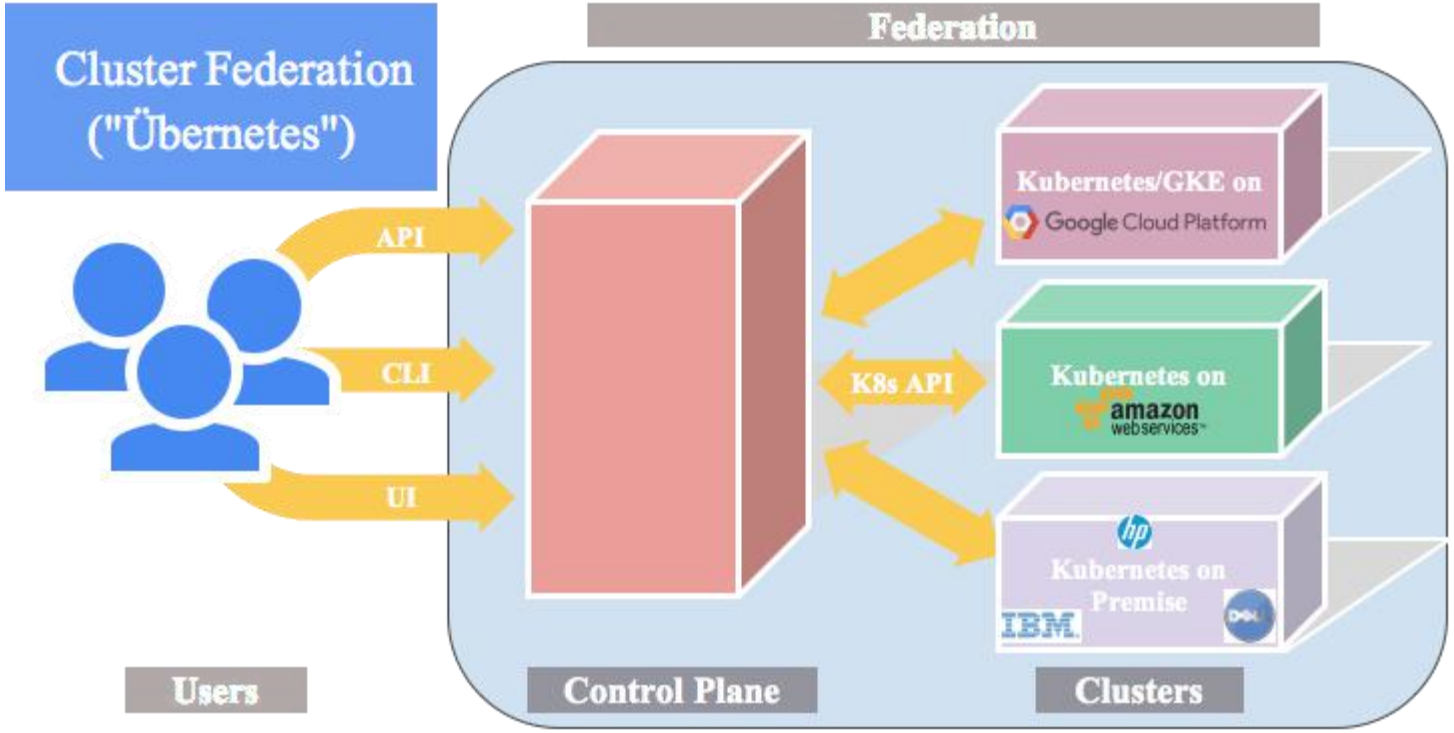
Formerly known as PetSet

Managing applications with state

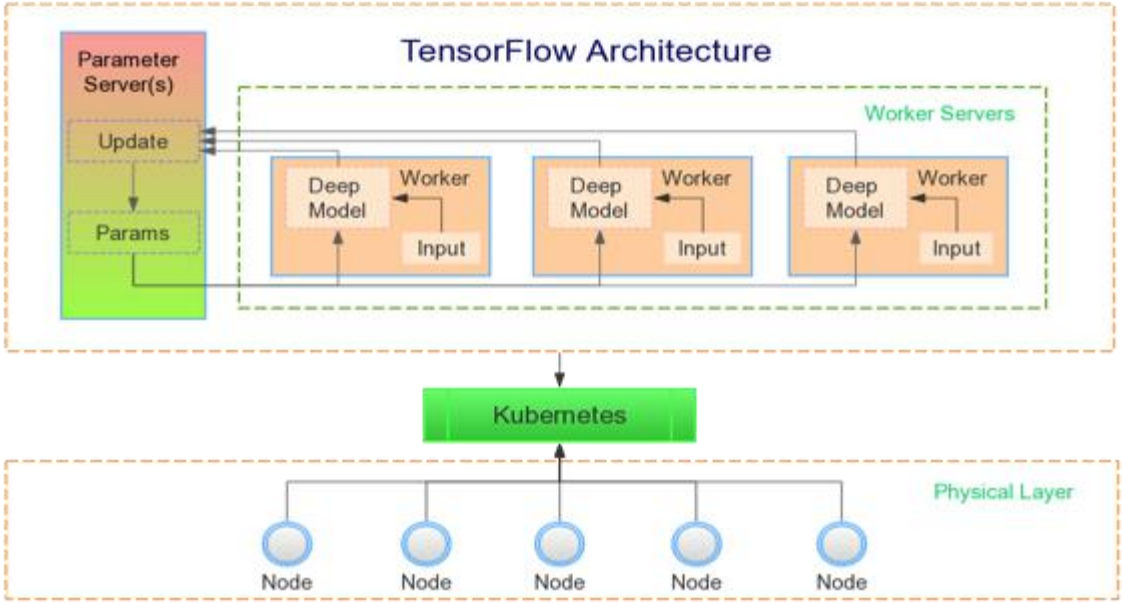
Example: PostgresSQL

- Node identity
- Replication connection
- Master connection
- Persistent Storage

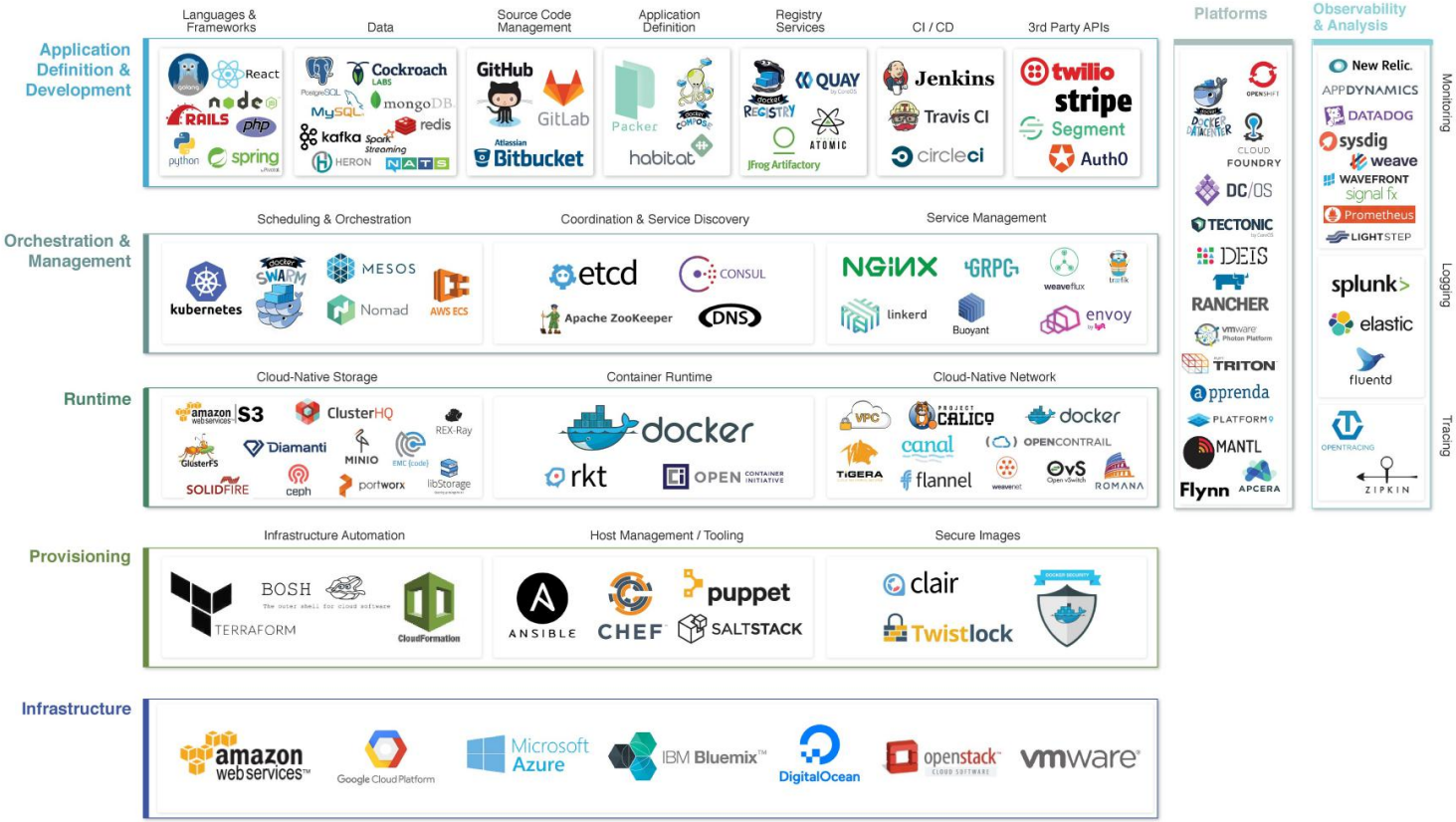
多区域集群联邦


Cluster Federation
("Übernetes")
Federation
API
CLI
UI
K8s API
Kubernetes/GKE on
Google Cloud Platform
Kubernetes on
amazon
web services
Kubernetes on
Premise
IBM
Users
Control Plane
Clusters

更好的大数据、深度学习支持



Cloud Native Landscape (US)



Kubernetes生产使用的配套工具

镜像仓库

- Harbor: https://github.com/caicloud/harbor

持续集成、持续发布

- Cyclone: https://github.com/caicloud/cyclone

复杂应用的管理

- Helm: https://github.com/kubernetes/helm
- Operator: https://coreos.com/blog/introducing-operators.html

集群即服务

容器圈的展望

集群管理会成为容器圈的核心

- 也成为了兵家必争之地

容器标准化

- Docker与Google、CNCF的恩怨
- runC, CRI-O, rktlet
- CNI vs CNM

集群与底层runtime的解耦合