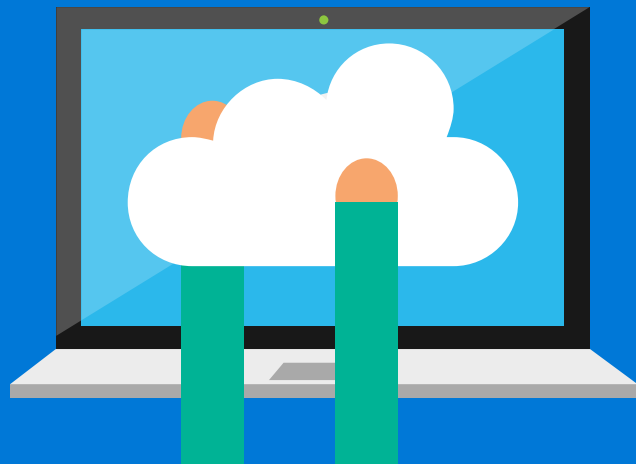




【DevOps云端开发训练营】

1.2 - Docker 容器编排平台与Azure 容器服务(ACS)

日程

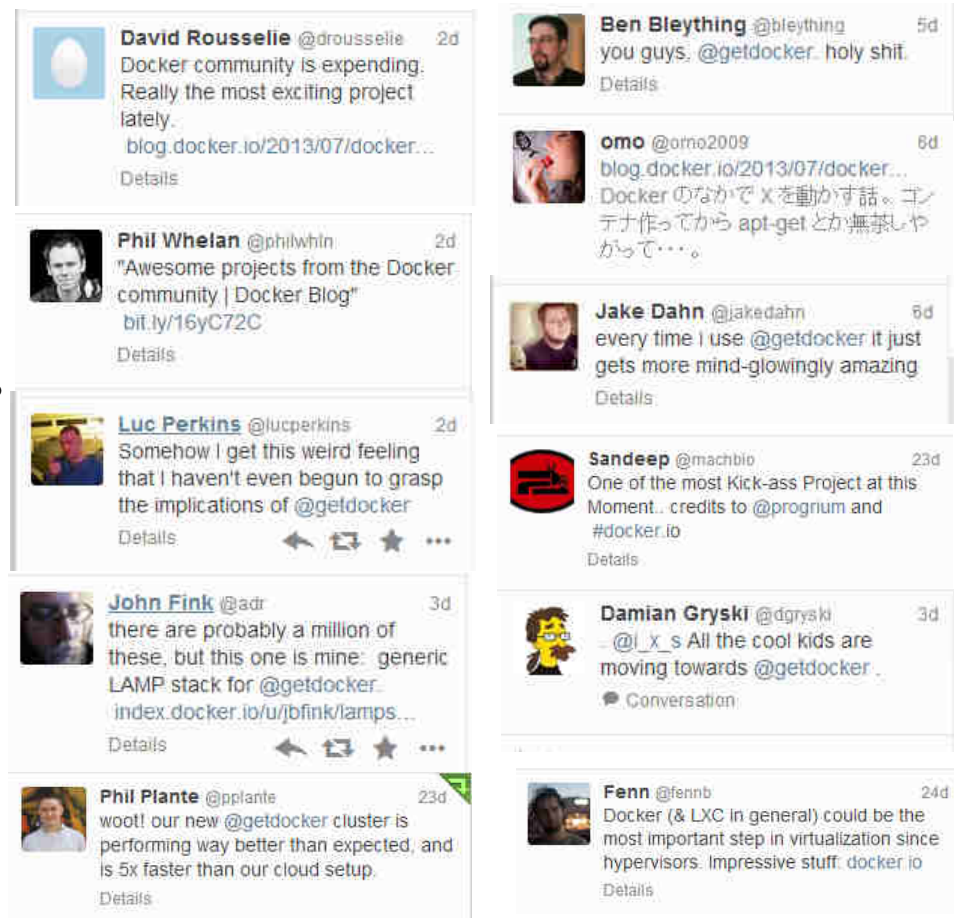


- Docker和容器技术概述？
- Docker对DevOps的价值
- Docker 工作机制
- 容器编排平台概述
- Azure 容器服务 (ACS)

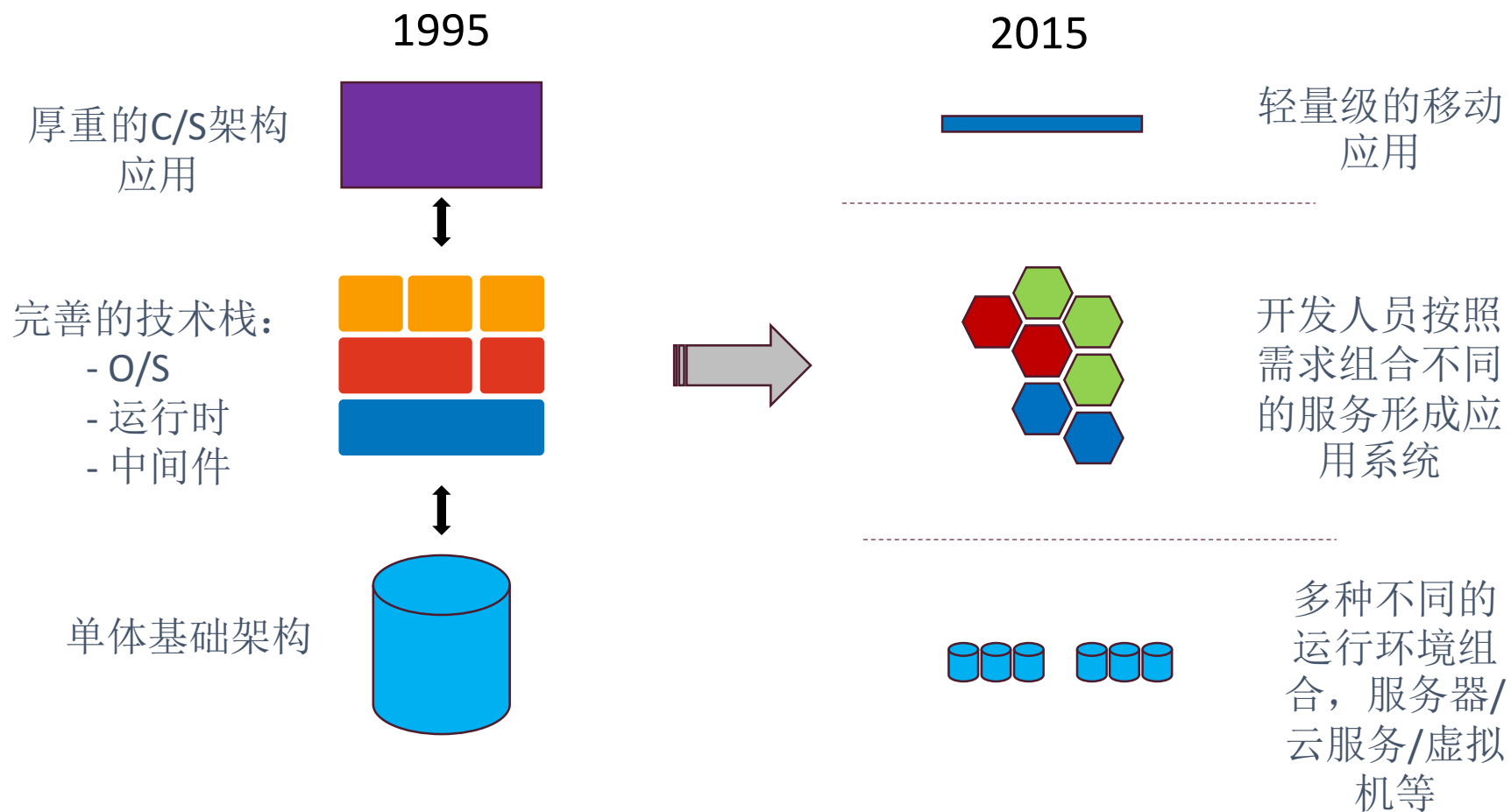
发布仅4个月即获得了极大关注

在发布4个月的时间里

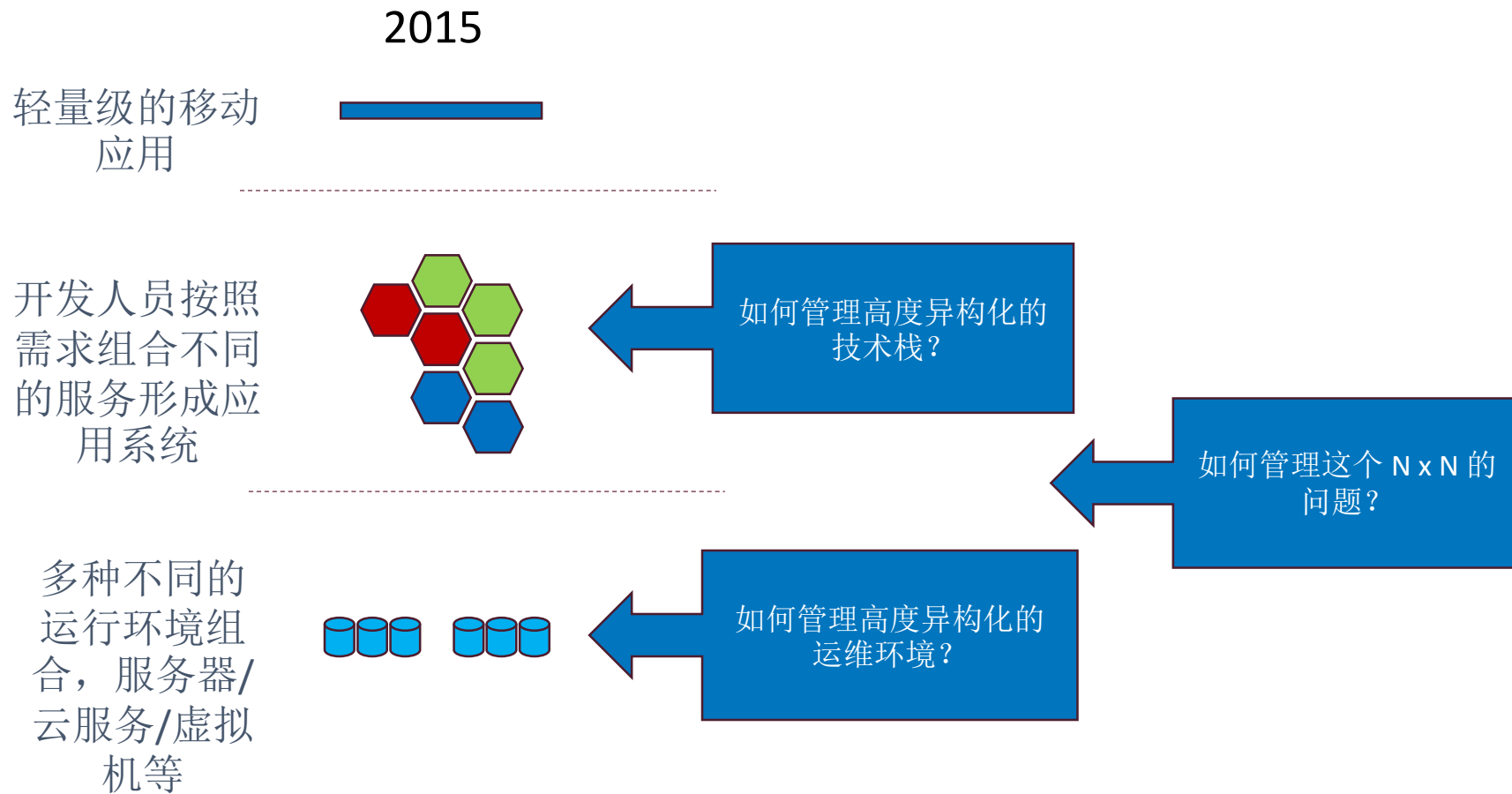
- 下载量就超过50000次，
 - github上收到超过4000个star，
 - 涌现了超过100个贡献者，
 - 并且有超过150个项目和超过1000个产品开始使用docker。
- 1年之内，
- RedHat和AWS就宣布为Docker提供官方支持
 - Docker自己的CEO都劝说全球的开发暂时不要将Docker用于生产环境。
 - 2014年6月，Docker发布了1.0版本，这时Docker的下载量已经超过275万，到今天这个数字已经超过了10亿。



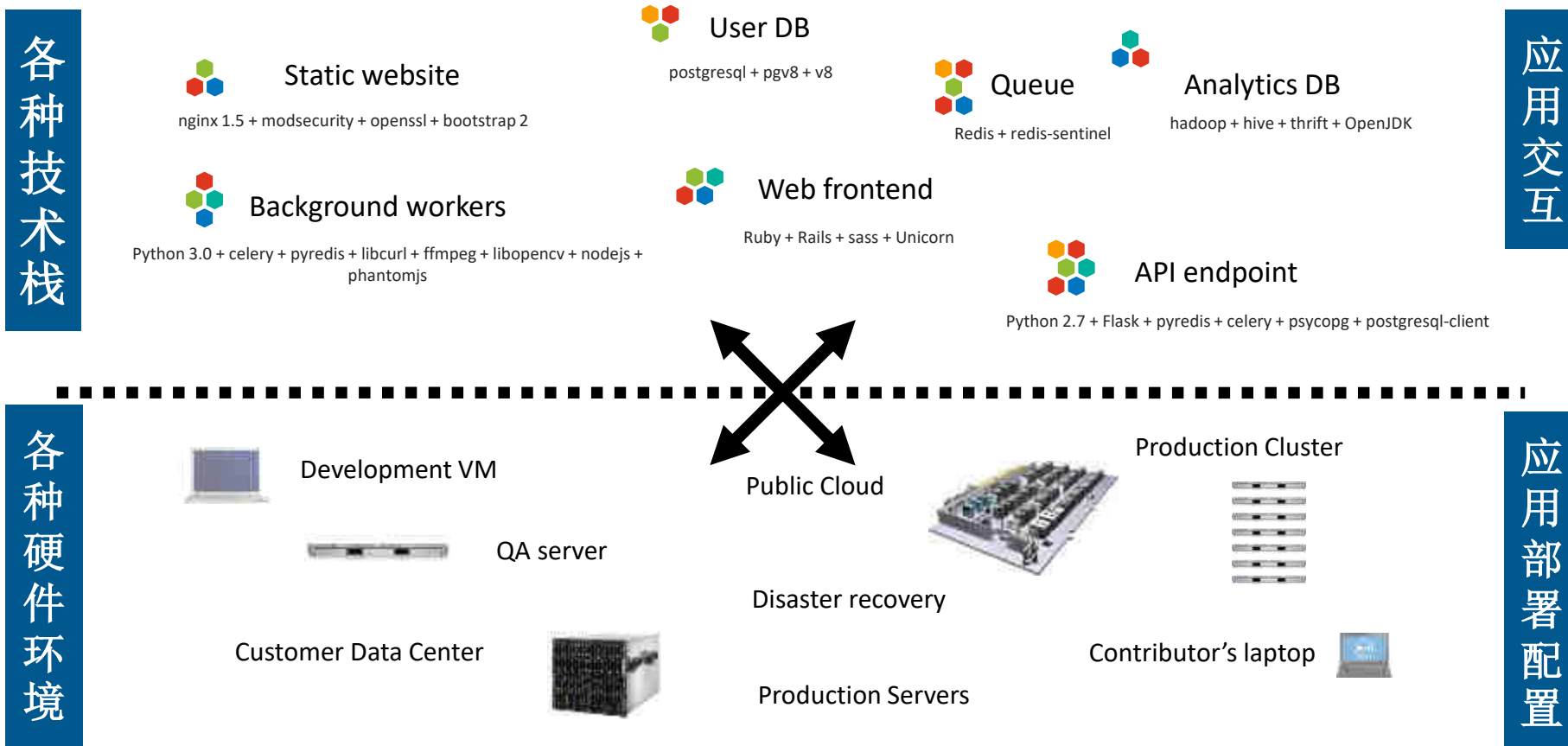
市场变化：IT运维的演化












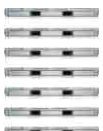



挑战



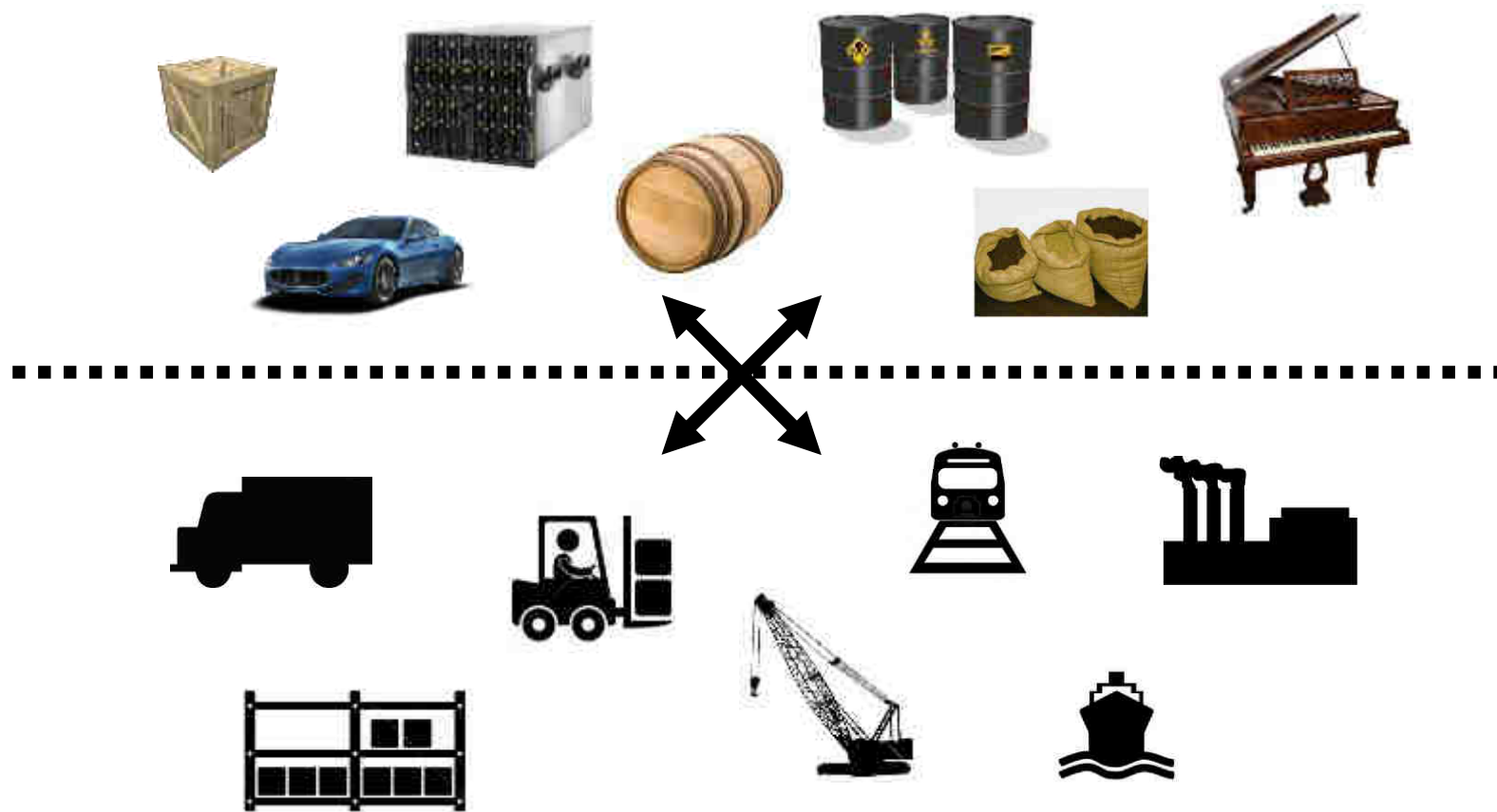
挑战










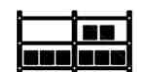





问题变成了N X N的问题矩阵！

	Static website	?	?	?	?	?	?	?
	Web frontend	?	?	?	?	?	?	?
	Background workers	?	?	?	?	?	?	?
	User DB	?	?	?	?	?	?	?
	Analytics DB	?	?	?	?	?	?	?
	Queue	?	?	?	?	?	?	?
		Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
								

1960年以前的运输行业




















































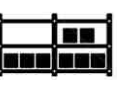





同样是 N×N 的矩阵问题！

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
							

解决方案: 集装箱



完美的解决了 NXN 矩阵问题...

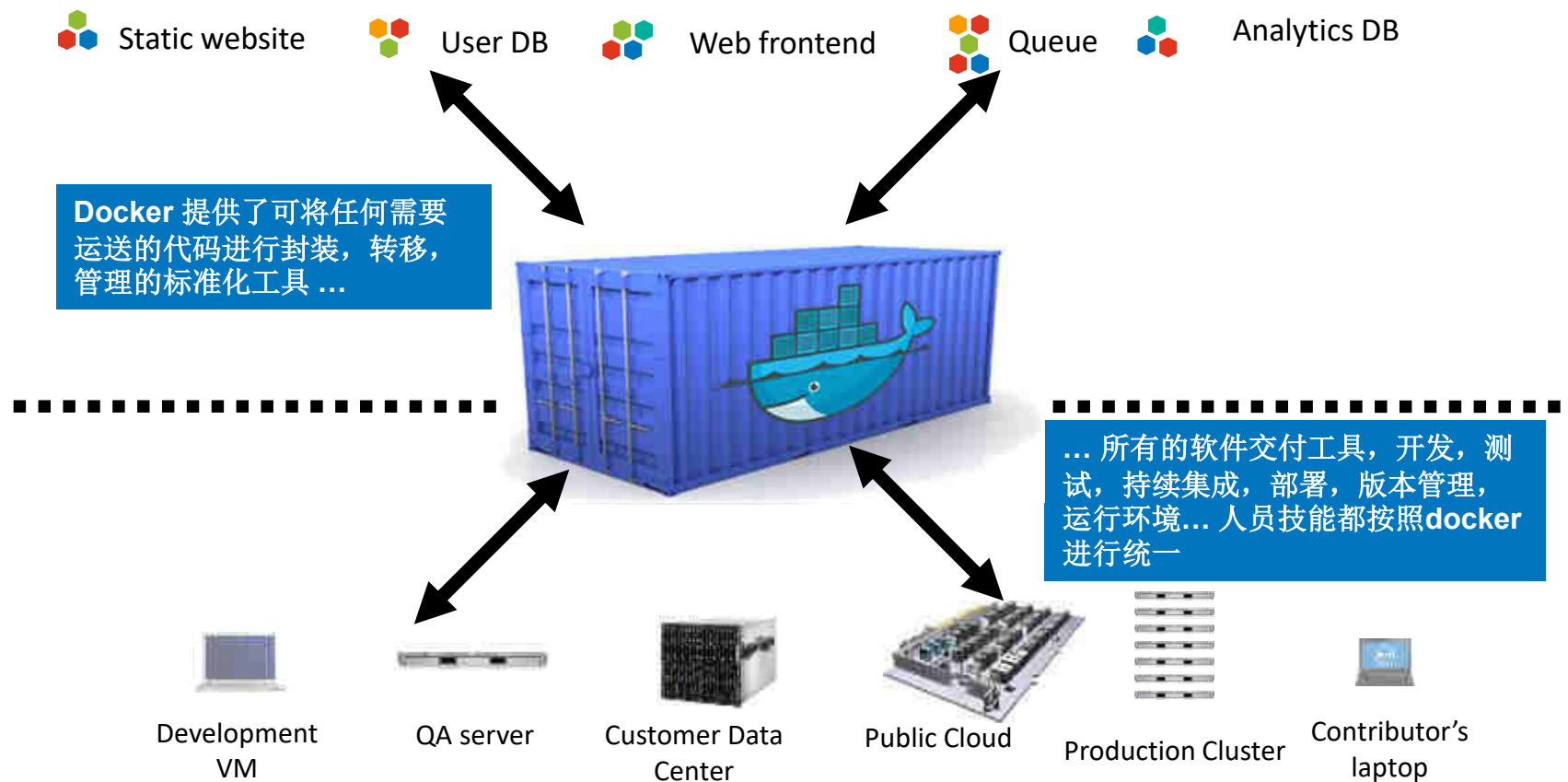
							
							
							
							
							
							
							

同时也创造出了全新的集装箱运输生态系统

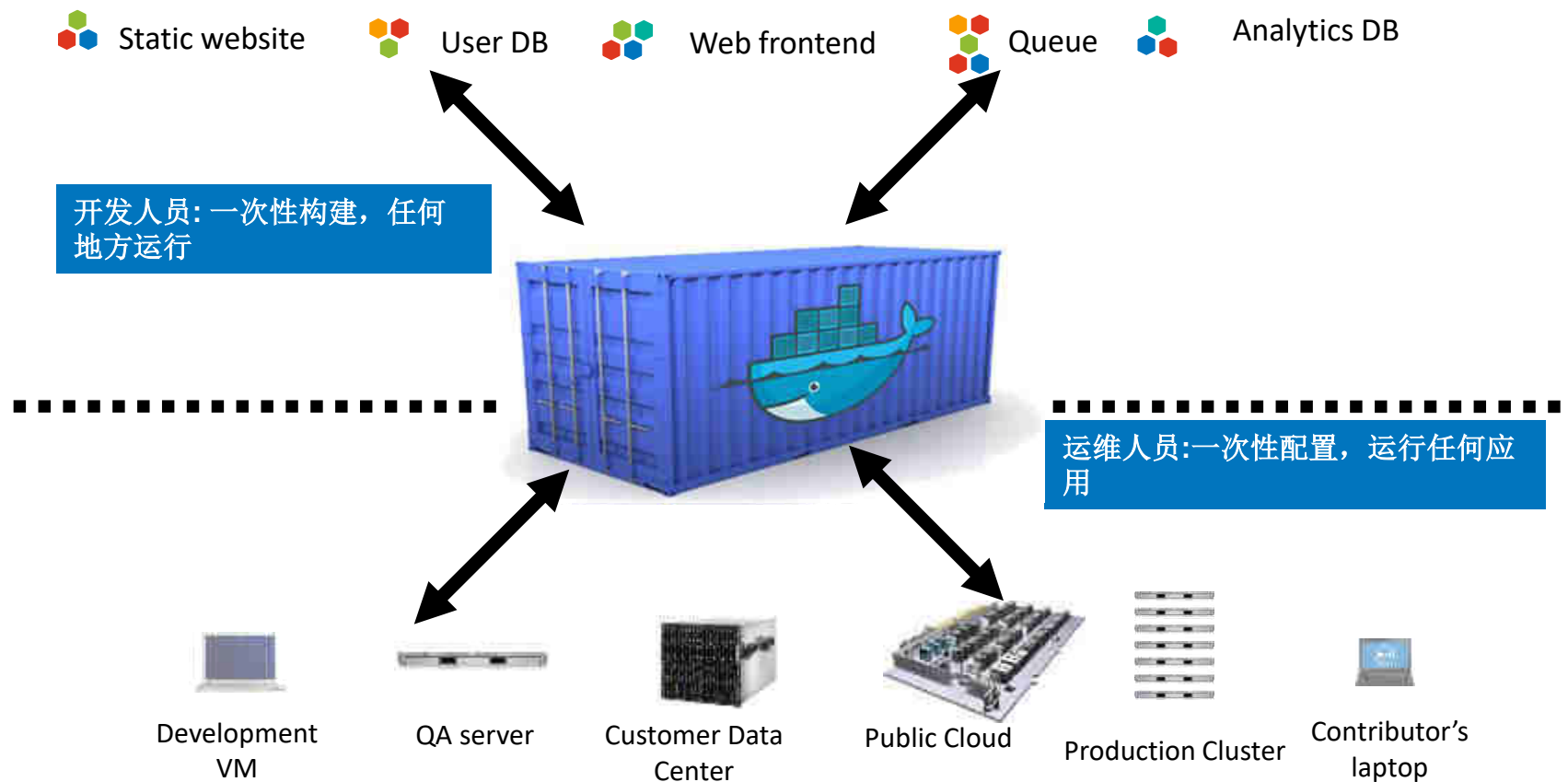


- 90% 的货物通过集装箱进行运输
 - 通过降低装载/卸载和运送的复杂度大大降低了运输成本
 - 大大减少了货物丢失和损坏的情况
 - 同时降低了货品上市时的售价，输入成本在售价中的比例大大降低 (从 >25% 降低至 <3%)
- 促进了全球化进程
- 每年有超过5000 艘货轮和超过20亿的集装箱被运送

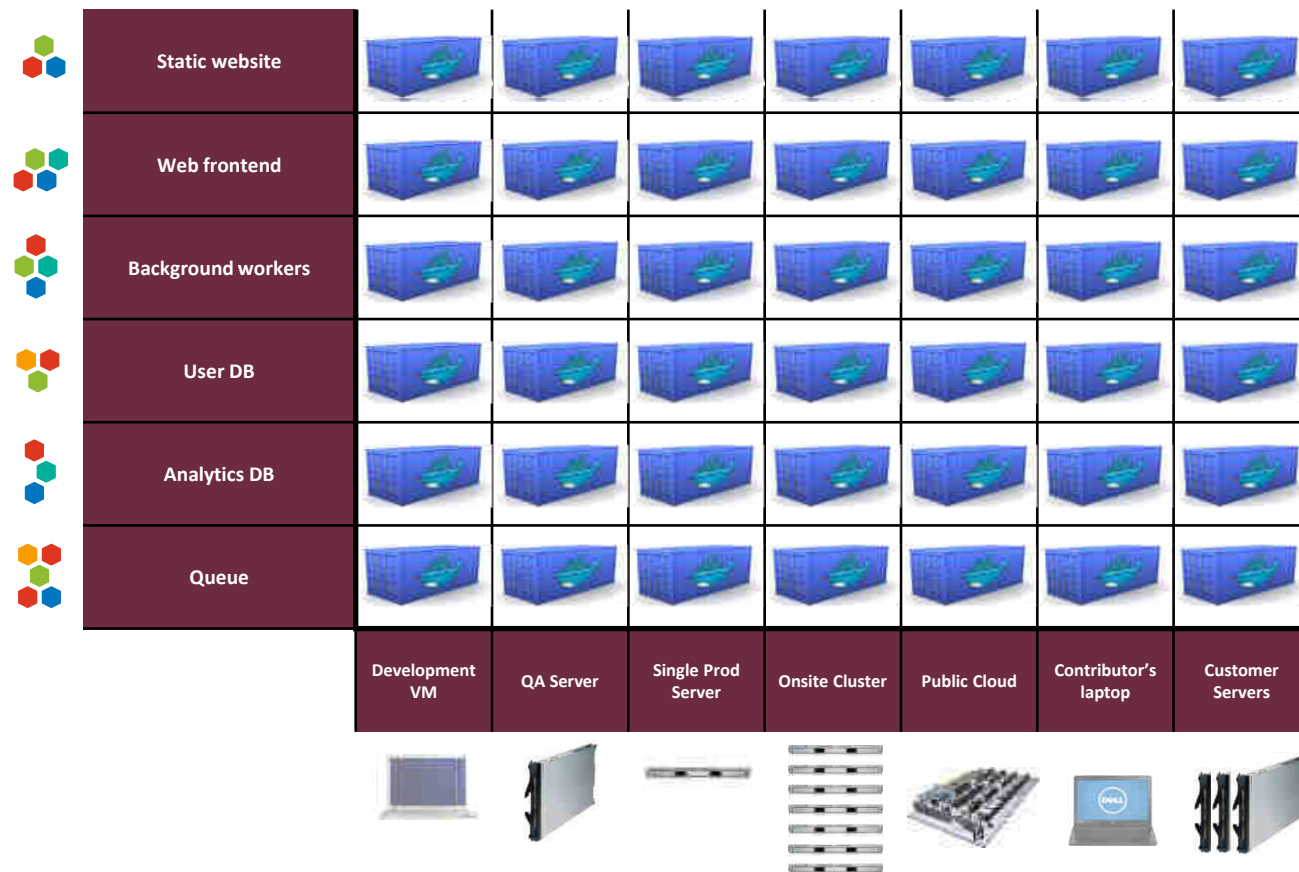
Docker：代码集装箱装卸工



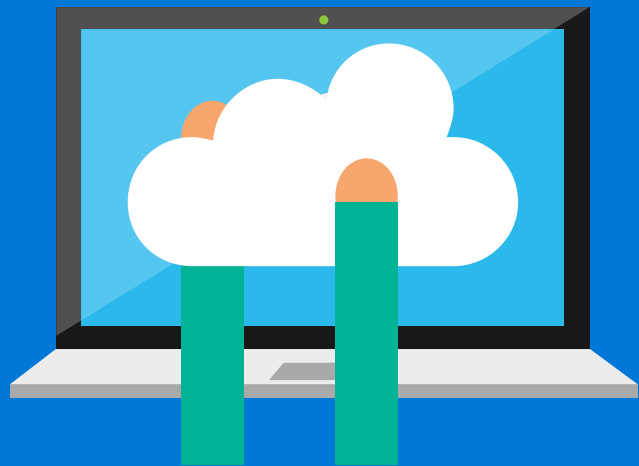
更简单直接的说法



Docker 解决了 NXN 矩阵的难题



日程



- Docker和容器技术概述?
- Docker对DevOps的价值
- Docker 工作机制
- 容器编排平台概述
- Azure 容器服务 (ACS)

对DevOps的价值

• 一次构建，多处运行

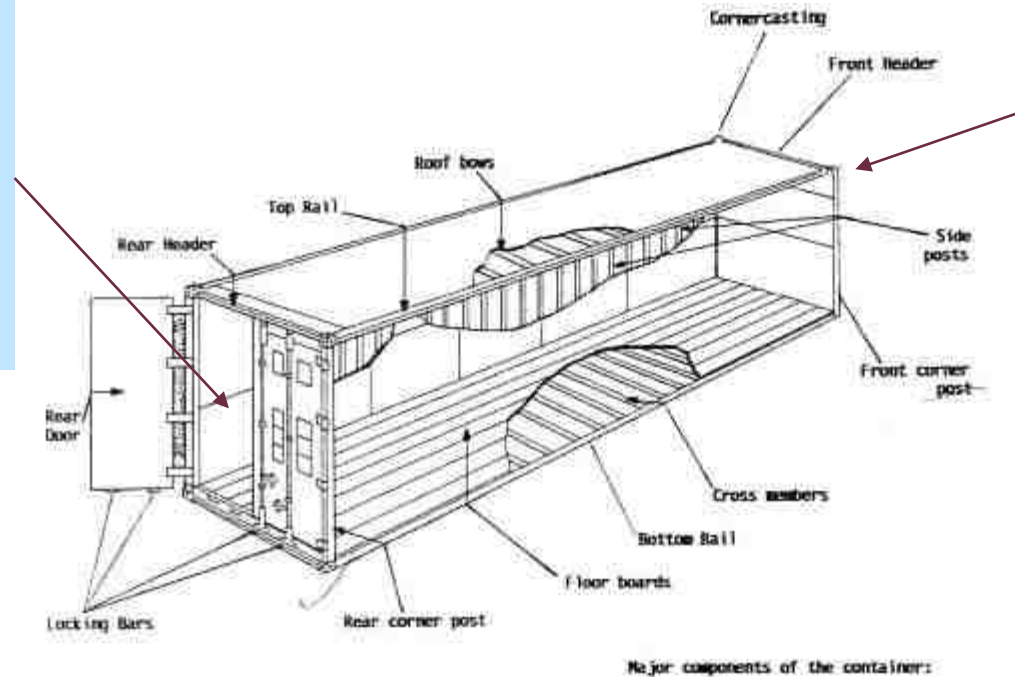
- 为应用提供了干净的，环境独立的，可迁移的运行平台
- 在多次部署中不必担心依赖，包和其他环境相关的配置
- 应用运行于独立的隔离的容器中，因此可以同时运行多个不同版本的库和依赖环境，而不用担心他们互相影响
- 自动化测试，集成，打包过程；全部可以通过简单的可版本化的脚本实现
- 降低与不同应用运行平台的兼容性问题
- 享受VM所提供的隔离性，快照等能力，同时又不被笨重的VM所拖累

• 配置一次，运行任何应用

- 让应用生命周期管理变得更加高效，统一并可复制
- 提升开发人员的代码质量
- 消除开发，测试，生产和客户定制化环境的差异性
- 为不同职能/技能的人员各司其职提供了条件
- 大大提升了持续集成和持续部署的可靠性，速度和可复制性
- 应为容器非常轻量，VM所存在的性能，成本，部署和可迁移性问题都迎刃而解

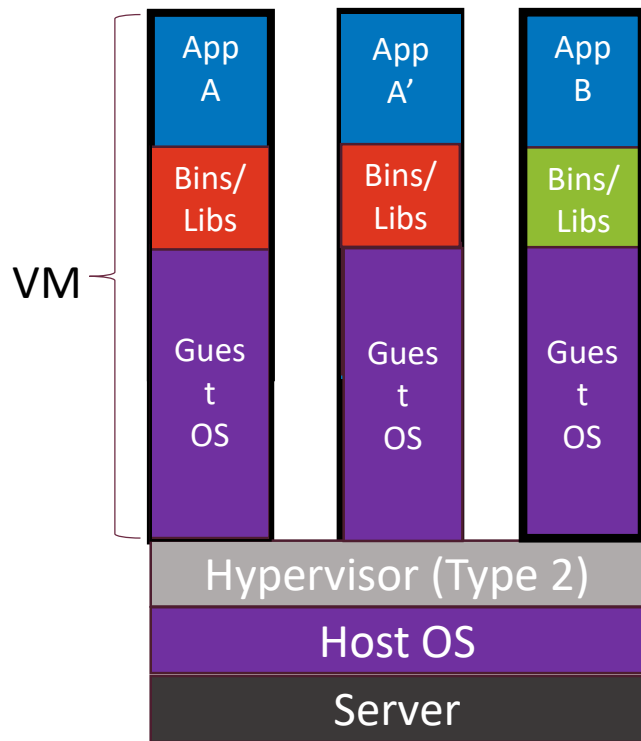
Why it works—separation of concerns

- Dan the Developer
 - Worries about what's "inside" the container
 - His code
 - His Libraries
 - His Package Manager
 - His Apps
 - His Data
 - All Linux servers look the same



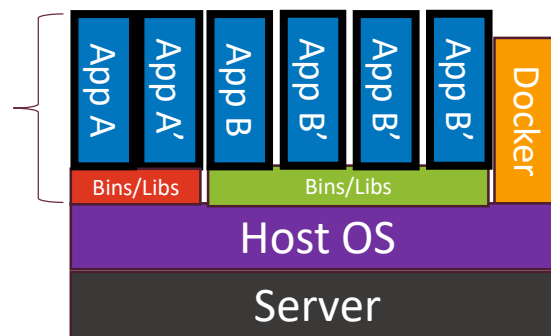
- Oscar the Ops Guy
 - Worries about what's "outside" the container
 - Logging
 - Remote access
 - Monitoring
 - Network config
 - All containers start, stop, copy, attach, migrate, etc. the same way

容器 vs. VMs

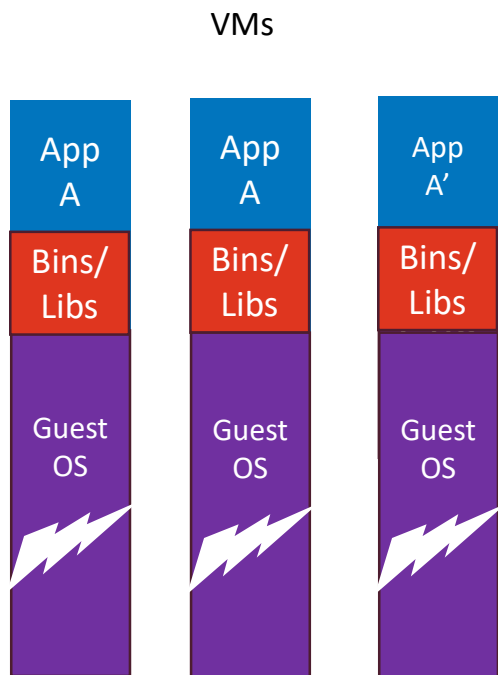


容器是隔离的，但是共享操作系统核心

容器



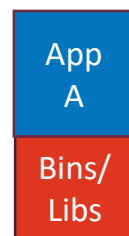
为什么Docker容器可以很轻量？



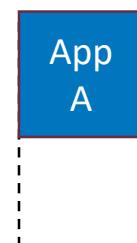
VMs

Every app, every copy of an app, and every slight modification of the app requires a new virtual server

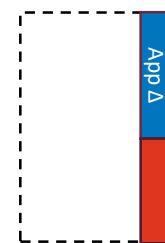
容器



原始应用
(不含操作系统，底层核心) 与原始应用共享文件系统



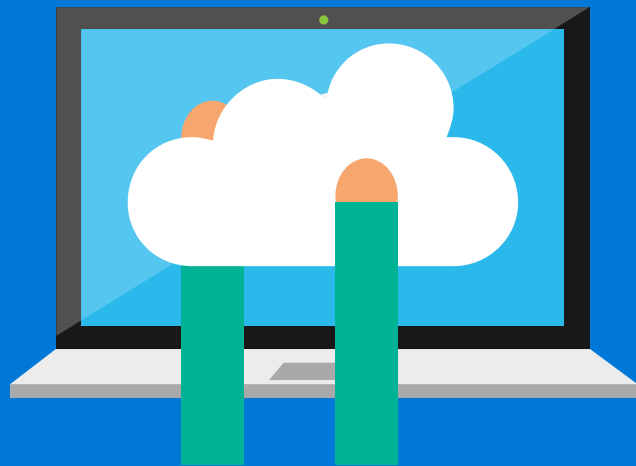
复制的容器



修改过的应用

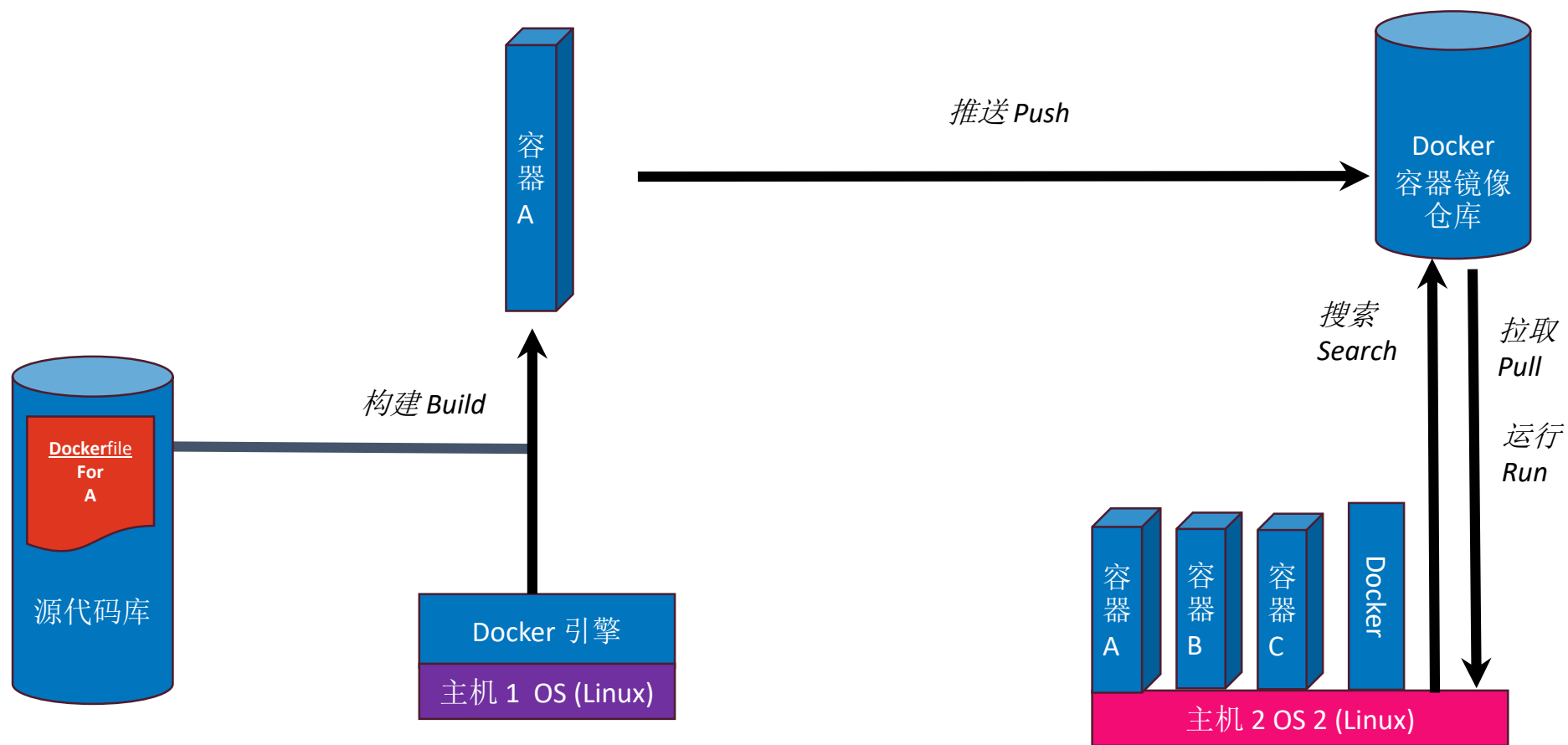
容期间未修改部分共享文件系统

日程

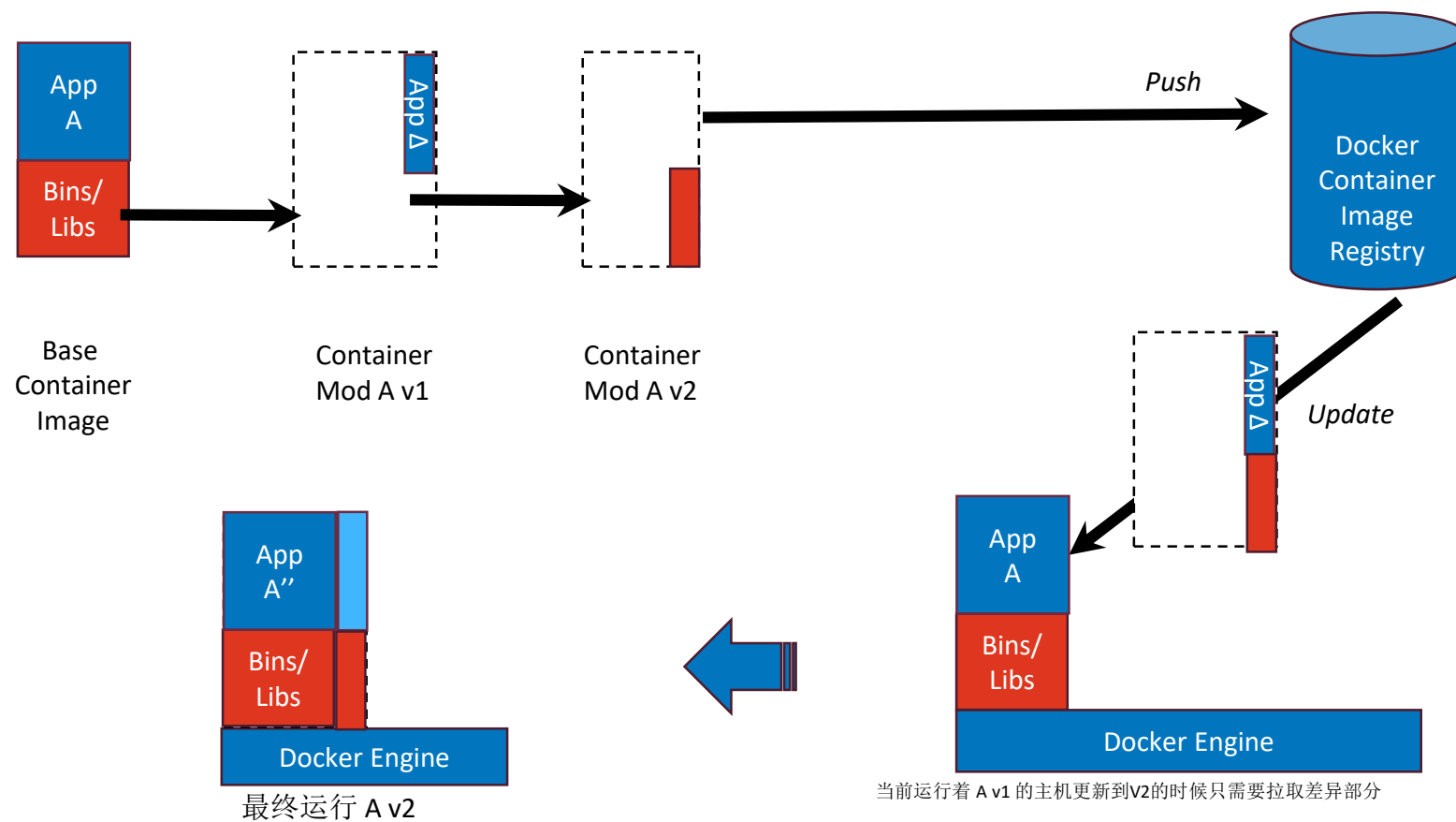


- Docker和容器技术概述?
- Docker对DevOps的价值
- Docker 工作机制
- 容器编排平台概述
- Azure 容器服务 (ACS)

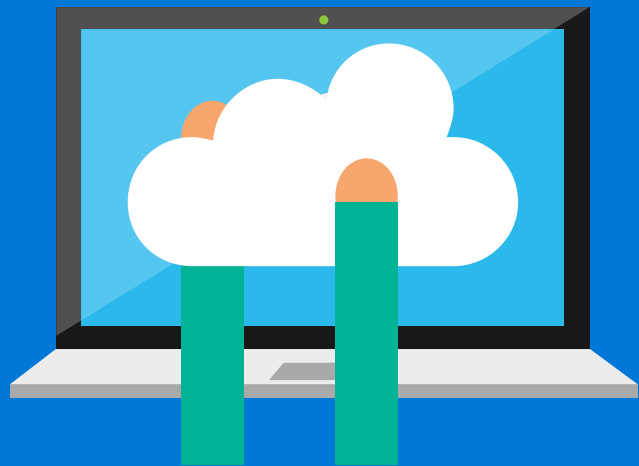
Docker发布系统工作机制



修改和更新的场景



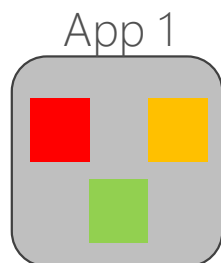
日程



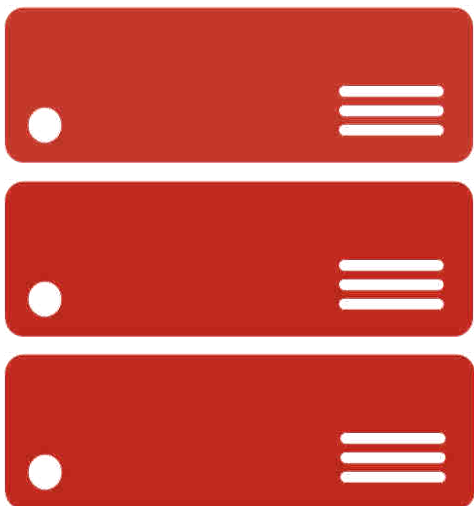
- Docker和容器技术概述?
- Docker对DevOps的价值
- Docker 工作机制
- 容器编排平台概述
- Azure 容器服务 (ACS)

单体应用架构

- 单体应用一般对应到某一业务领域所需要的各种功能和能力，并按照功能进行分层设计，如：web，业务逻辑层和数据层。

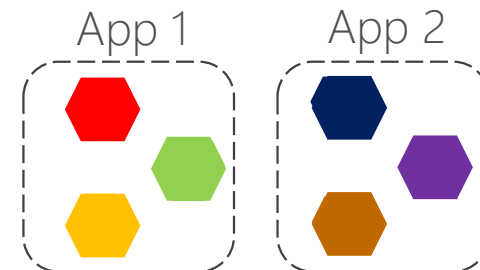


- 通过多套服务/虚拟机/容器中整体复制的方式进行扩展

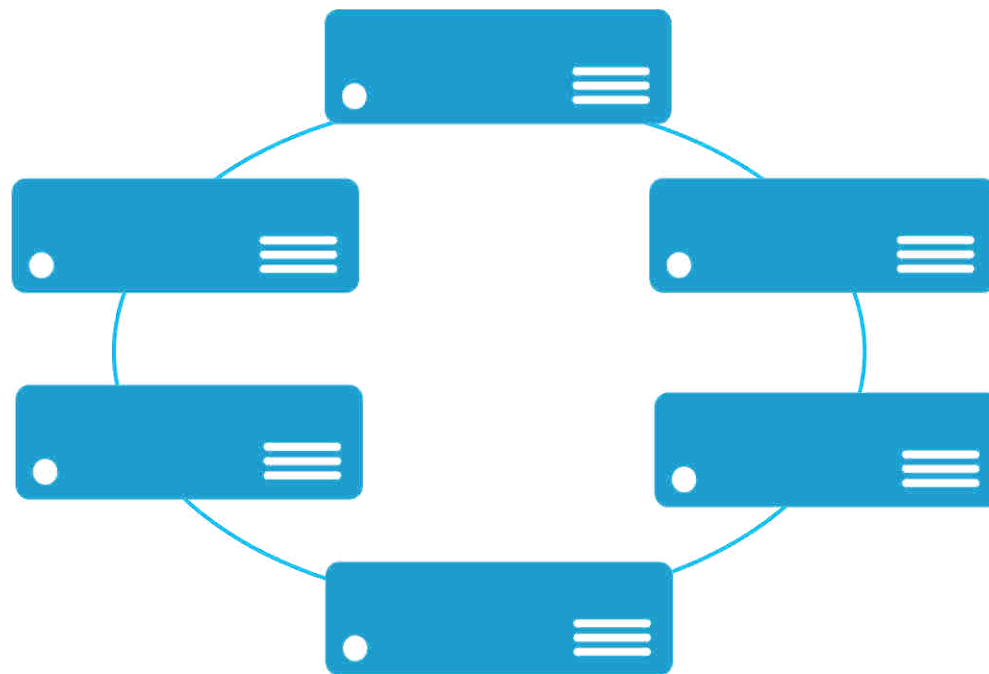


微服务应用架构

- 微服务应用根据功能拆分为不同的可以独立运行的服务中。

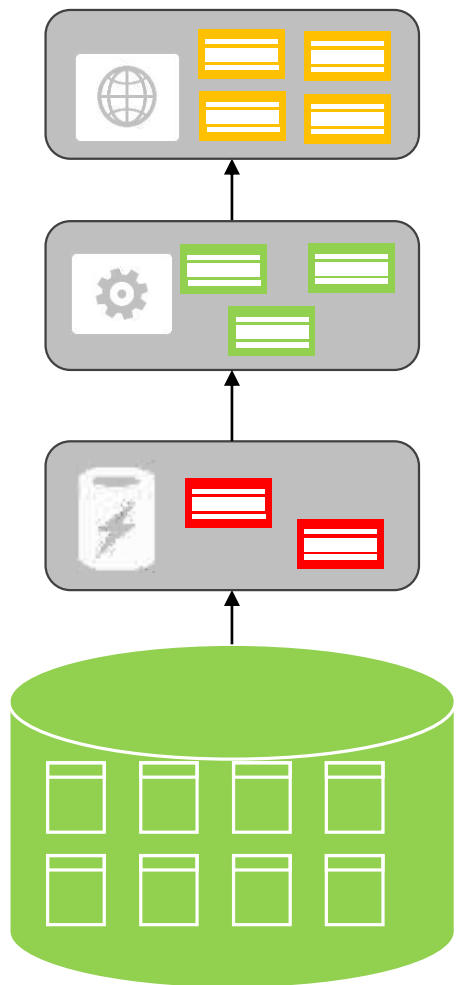


- 通过在服务器/虚拟机/容器中克隆服务的方式继续进行扩展



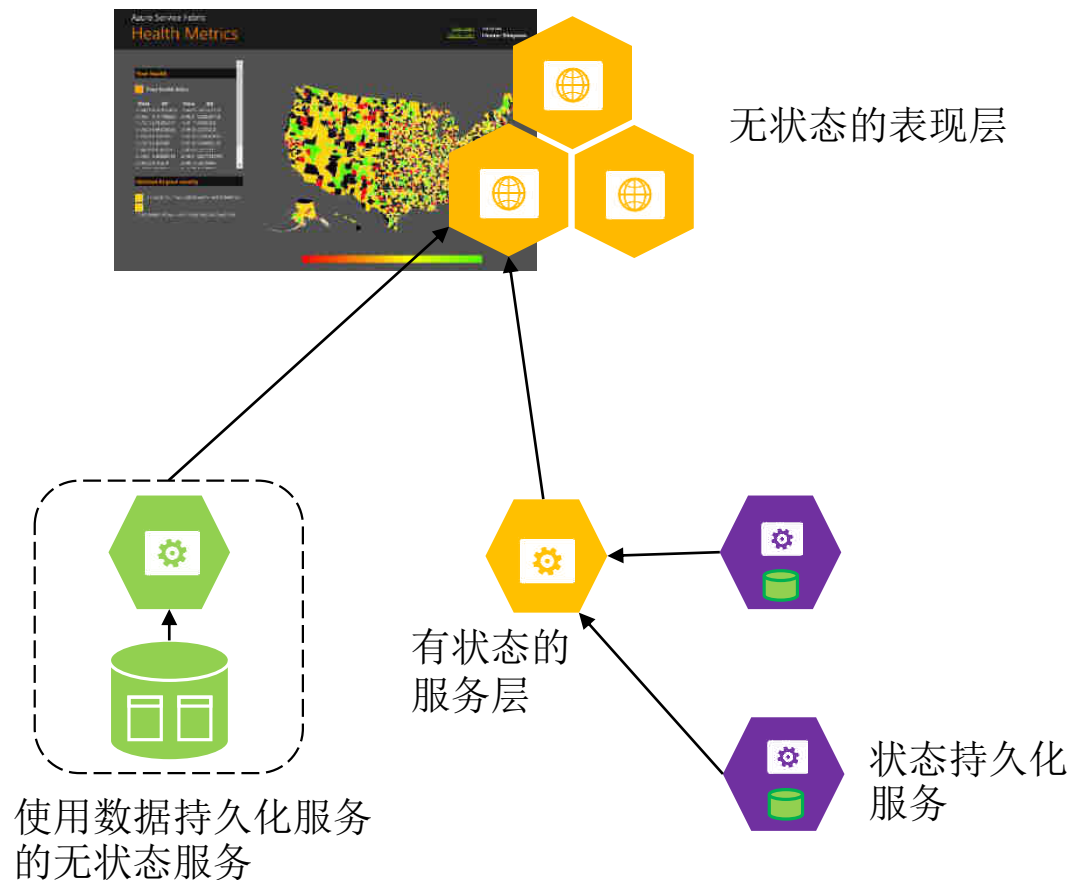
单体架构中的状态持久化

- 使用独立的数据库持久化数据
- 各应用层可以使用不同的技术实现



微服务架构中的状态持久化

- 独立服务组成服务网络
- 各个服务独立处理自己的状态持久化
- 各服务独立决定所使用的技术实现
- 不常用的数据被备份到远程存储



编排平台

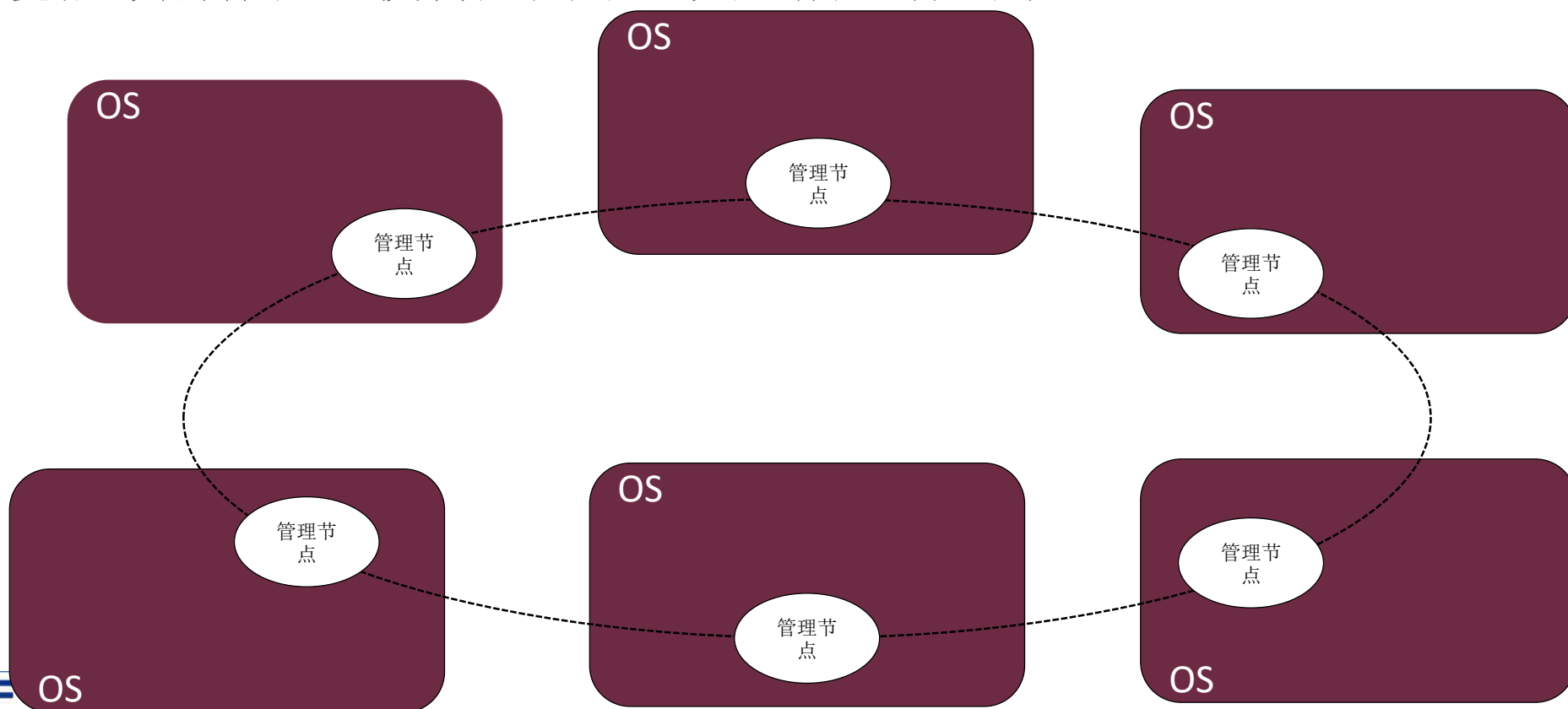
支持使用不同的开发语言和框架实现服务

容器编排平台

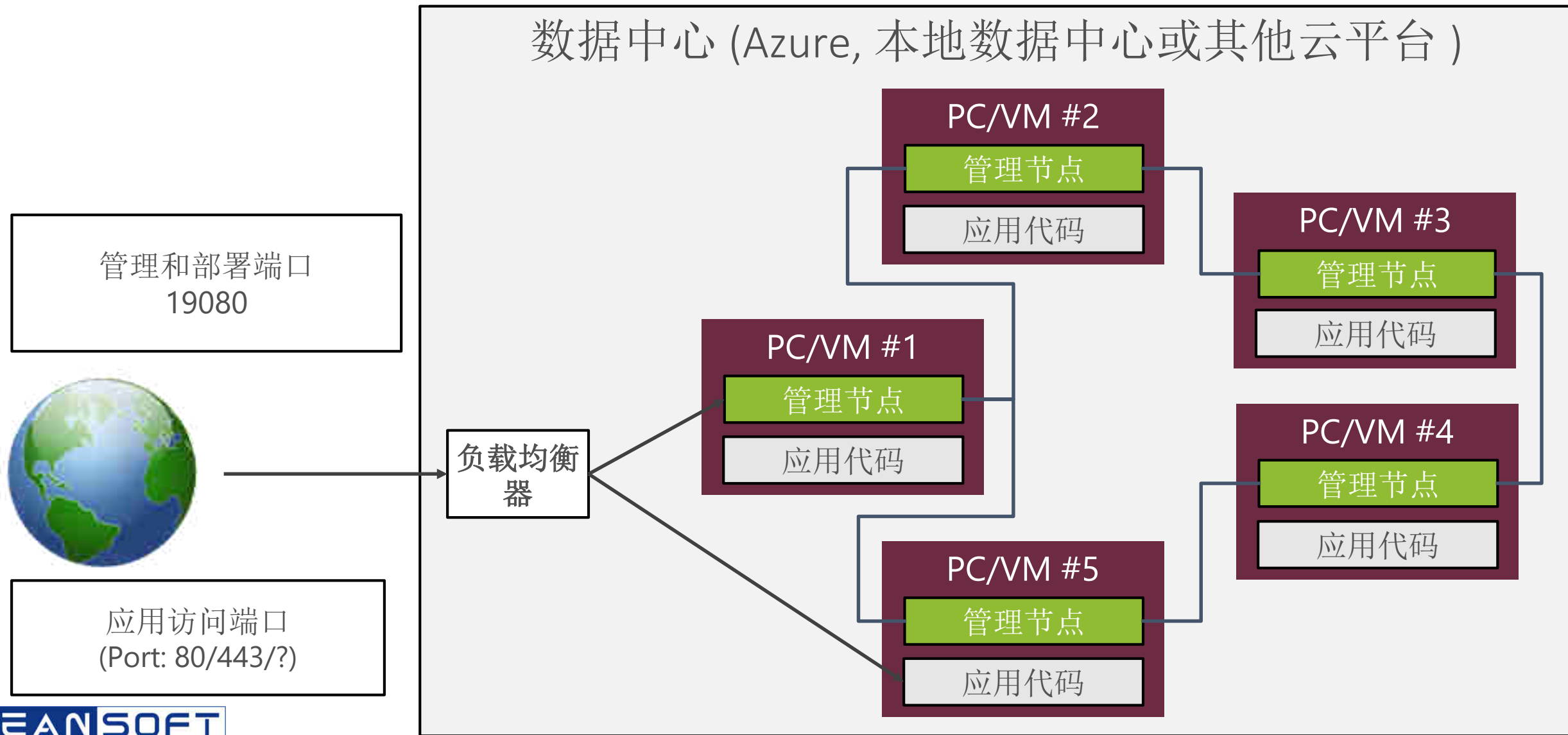
支持部署到不同的环境（公有云，私有云，独立数据中心，虚拟机和实体机）

编排平台的特性

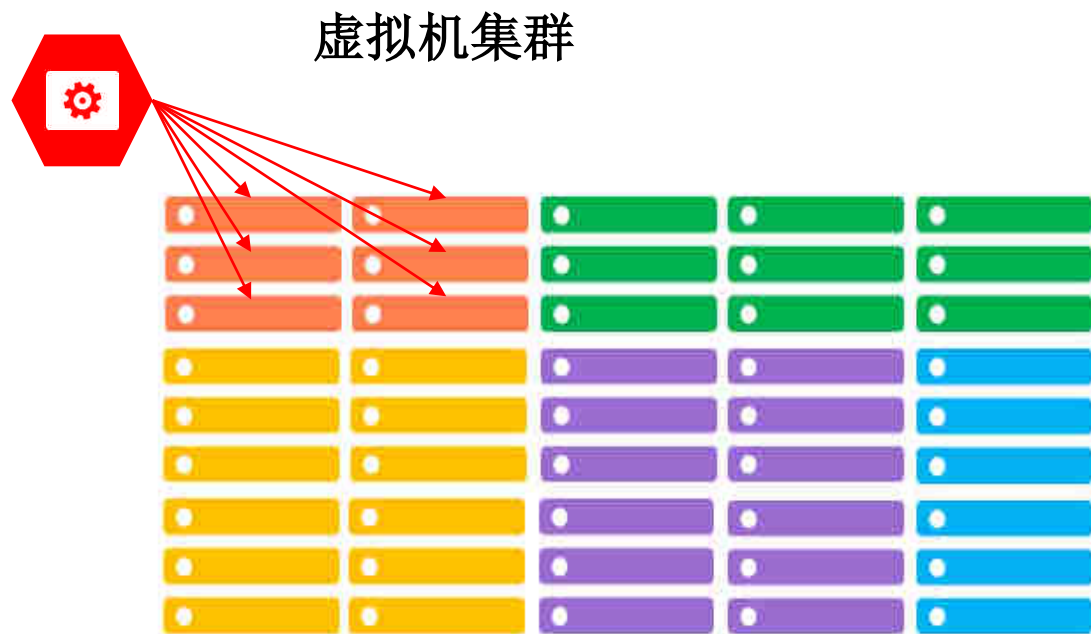
- 一组互相连接的操作系统（实体机/虚拟机/容器）形成一个整体的资源池
- 可以扩展到上千个节点，具备自我修复，扩展和收缩能力
- 作为环境抽象层存在，使得应用不必关心所运行的节点



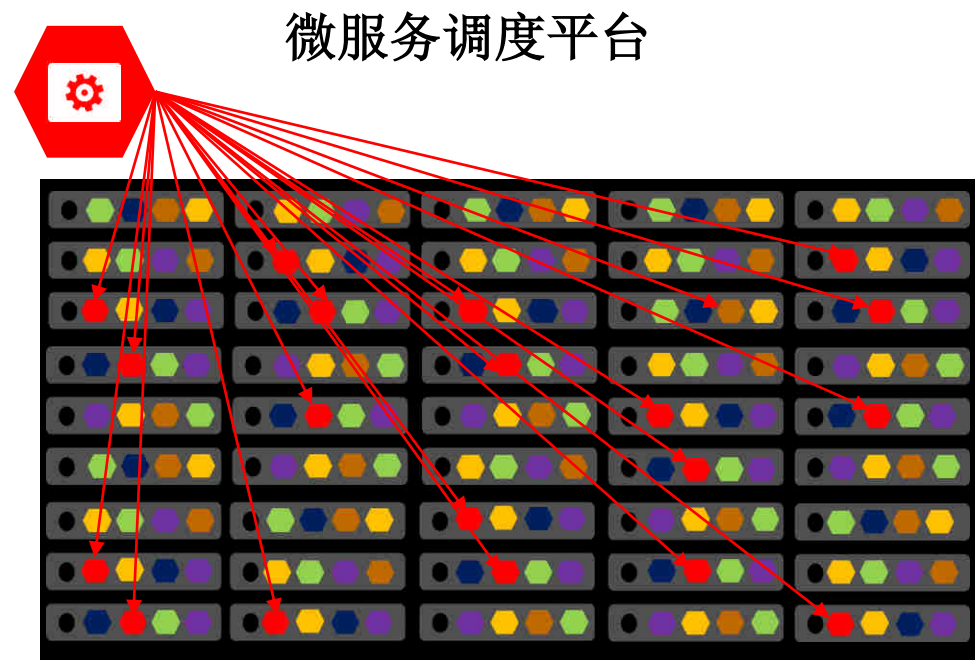
编排平台的工作方式



虚拟机集群 vs. 编排平台



- 一个VM一个角色
- 资源利用不均衡
- 密度低
- 部署和升级速度慢（绑定到VM）
- 扩展和故障恢复慢
- 容错能力有限



- 一个VM多个微服务
- 优化资源利用率 (可定制的策略)
- 高密度 (可定制)
- 快速部署和升级
- 以微服务为单位快速扩展
- 可定制的容错能力

四大编排平台

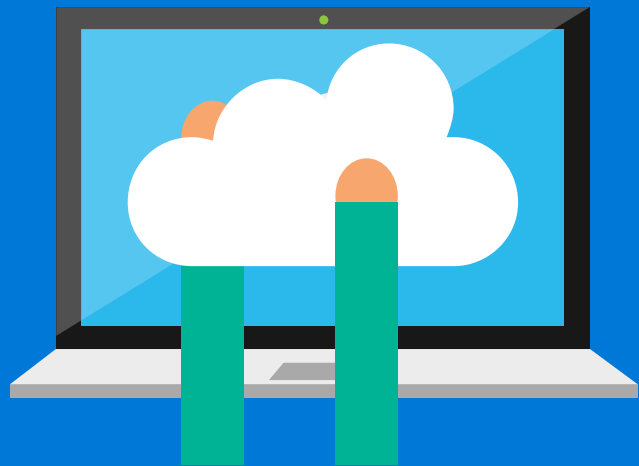
- Apache Mesos & Marathon (DC/OS)
- Google Kubernetes (k8s)
- Docker Swarm (Docker Datacenter)
- Microsoft Azure Service Fabric



编排平台

- 进一步抽象了计算资源（虚拟机/实体机）
- 针对微服务运维提供动态资源调度的Paas平台
- 跨云，跨操作系统，跨厂商
- 面向应用
- +容器：跨应用技术栈

日程



- Docker和容器技术概述?
- Docker对DevOps的价值
- Docker 工作机制
- 容器编排平台概述
- Azure 容器服务 (ACS)

Azure Open Source Container Platform

Enabling agility with containers in the cloud

Developers



Operations



Data analysts



Docker VM Extension for Azure

Easy and programmatic way to add Docker capabilities to your VMs



Open source container-based PaaS platforms in Azure

Container-ready application platforms that benefit from Azure's native partitioning, capacity management and high availability



Azure Container Service

Optimized container hosting in the cloud with familiar tooling and your choice of orchestrator



Azure Marketplace container partners

Partner solutions that address management challenges of containers



Microsoft Azure



Workload portability



Cross-cloud orchestration



Tools integration

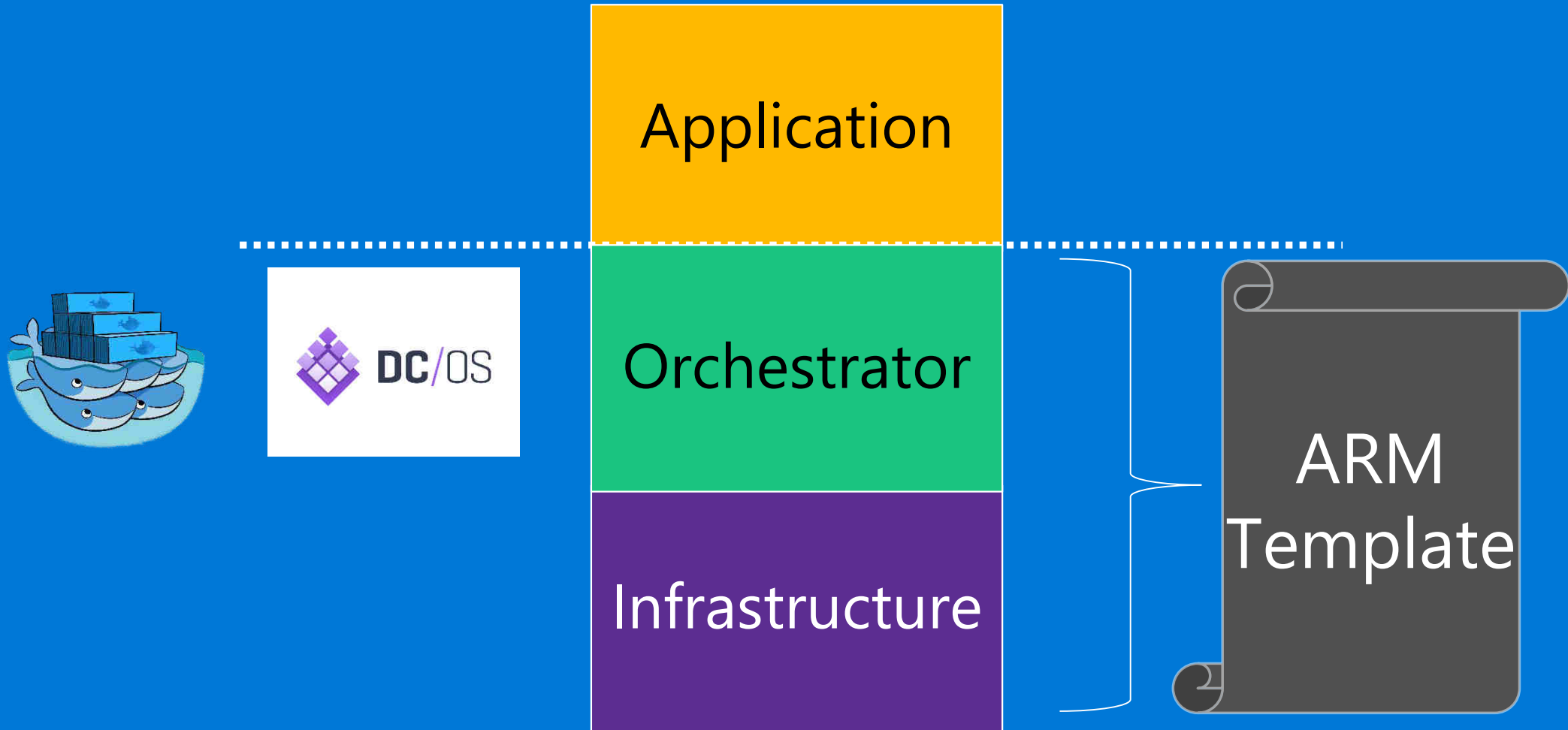
Windows Server (preview)

Azure Stack (roadmap)

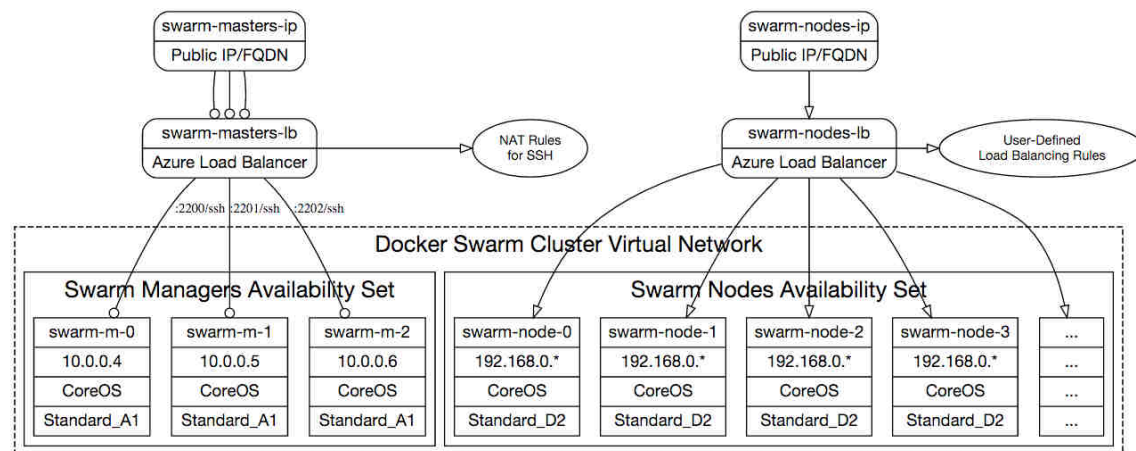


Your own platform

Azure Container Service Layers



Docker Swarm in Microsoft Azure



- **Swarm Manager**
 - 维护集群内节点状态
 - 调度服务部署，扩容和更新
- **Swarm Nodes**
 - 承担工作负载
- **Swarm Cluster 私有网络**
 - 将Swarm集群与Azure云中的其他环境隔离，形成独立的网络空间，可以自主创建子网并决定ip地址
- **Azure 负载均衡器**
 - 提供对外管理和访问端口，并引导流量到达应用节点

Swarm Mode 特性

- 集成在Docker内部的集群管理能力
- 去中心化设计
- 声明式服务模型
- 扩容缩容
- 理想状态维护
- 跨主机网络
- 服务发现
- 负载均衡
- 内建的安全性
- 滚动更新

LEANSOFT 简介

Nobody knows DevOps better than us!

英捷创软科技(北京)有限公司(LEANSOFT)是一家专注于软件工程，敏捷开发和DevOps领域产品开发和服务的解决方案提供商。

公司由15年 软件研发经验,资深ALM/DevOps专家创建并任公司首席架构师，至今已经为超过100家不同类型的客户提供过ALM解决方案的咨询和落地服务。

联系我们

📞 137 1126 4760

✉ info@lean-soft.cn

我们致力于为广大开发人员和企业提供最优化的DevOps/敏捷咨询服务和软件工程解决方案。

请扫描右侧二维码加入中国最大的DevOps社区【DevOpsHub】

社区网站 <https://devopshub.cn>

