



当Devops遇到容器

淡成

红帽软件

2017.4.19



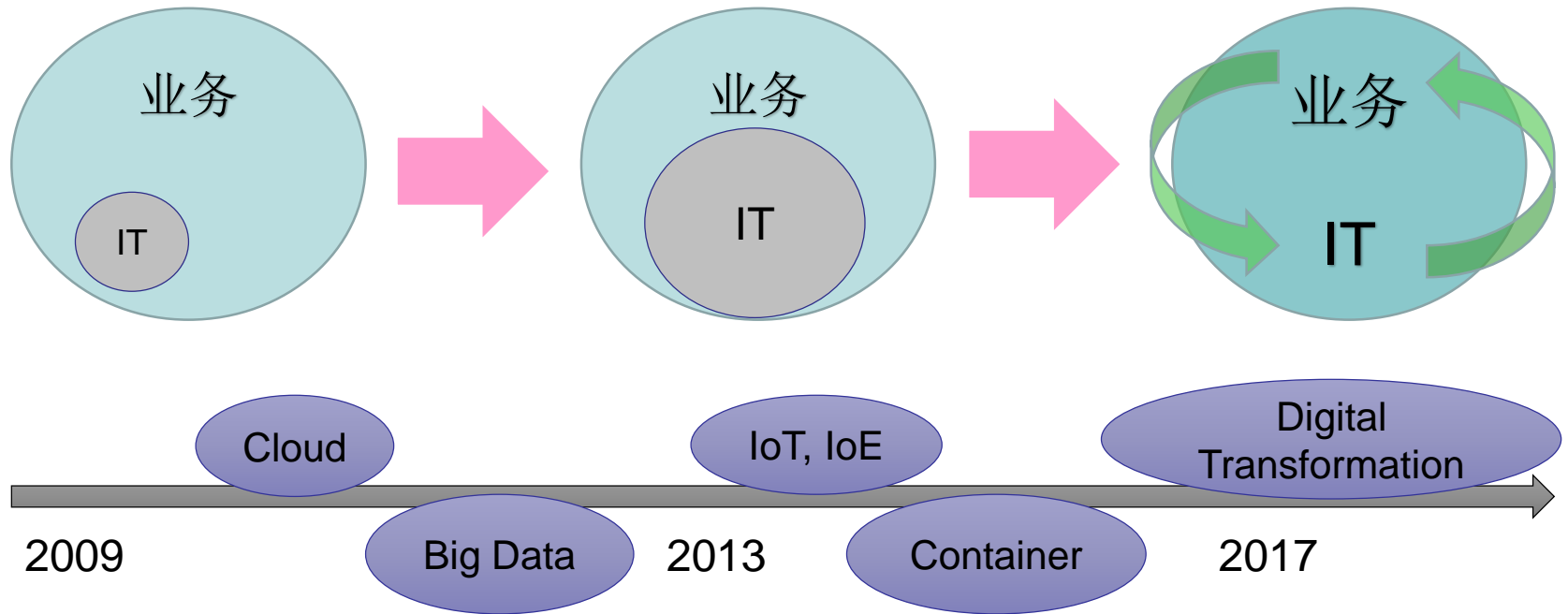
最初的Devops

10 deploys per day
Dev & ops cooperation at Flickr

John Allspaw & Paul Hammond
Velocity 2009



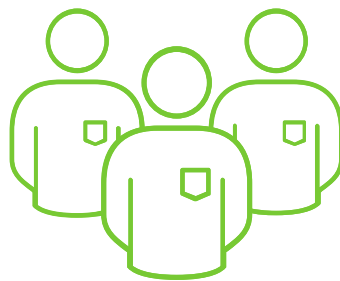
Devops发展的背景



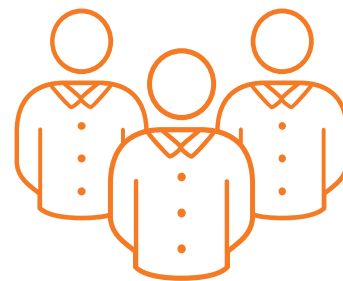
Devops的参与人员



产品经理
业务需求



开发人员
效率



运维人员
稳定性



敏捷

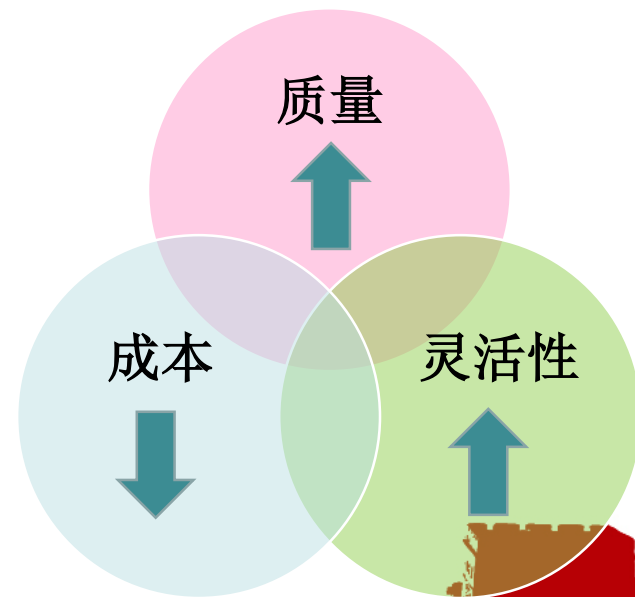
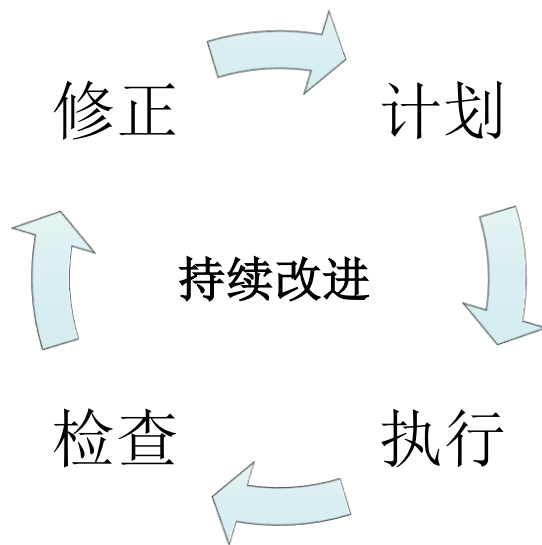
DevOps

实现业务目标



对IT部门的要求

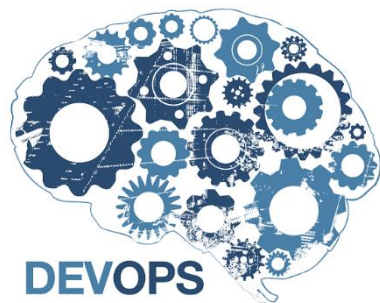
- 持续改进
 - 驱动业务
 - 适应业务变更
- 质量、成本、灵活性的结合



Devops的解决之道

- 人员、文化
 - ✓ 鼓励跨部门的合作 - Biz & Dev & Ops
 - ✓ 统一的考核目标
- 流程
 - ✓ 敏捷：通过迭代尽早发现和解决问题
 - ✓ DevOps 流水线：通过流水线管理不同阶段的交付
- ✓ 技术、架构、工具
 - ✓ CICD 流水线
 - ✓ 一切可编程、尽可能的自动化（减少人工干预）
 - ✓ 不可变基础设施，多种类部署模型（灰度、蓝绿）
 - ✓ 通过微服务获取灵活性、降低变更影响、培养Devops团队

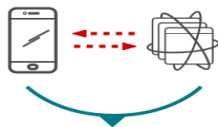




IT架构趋势

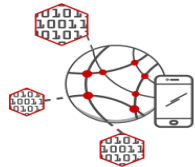
Application Architecture

Monolithic



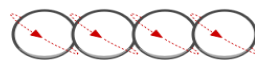
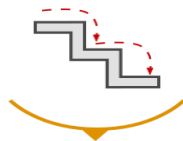
N-Tier

Microservices



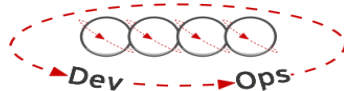
Development Process

Waterfall



Agile

DevOps



Application Infrastructure

Datacenter

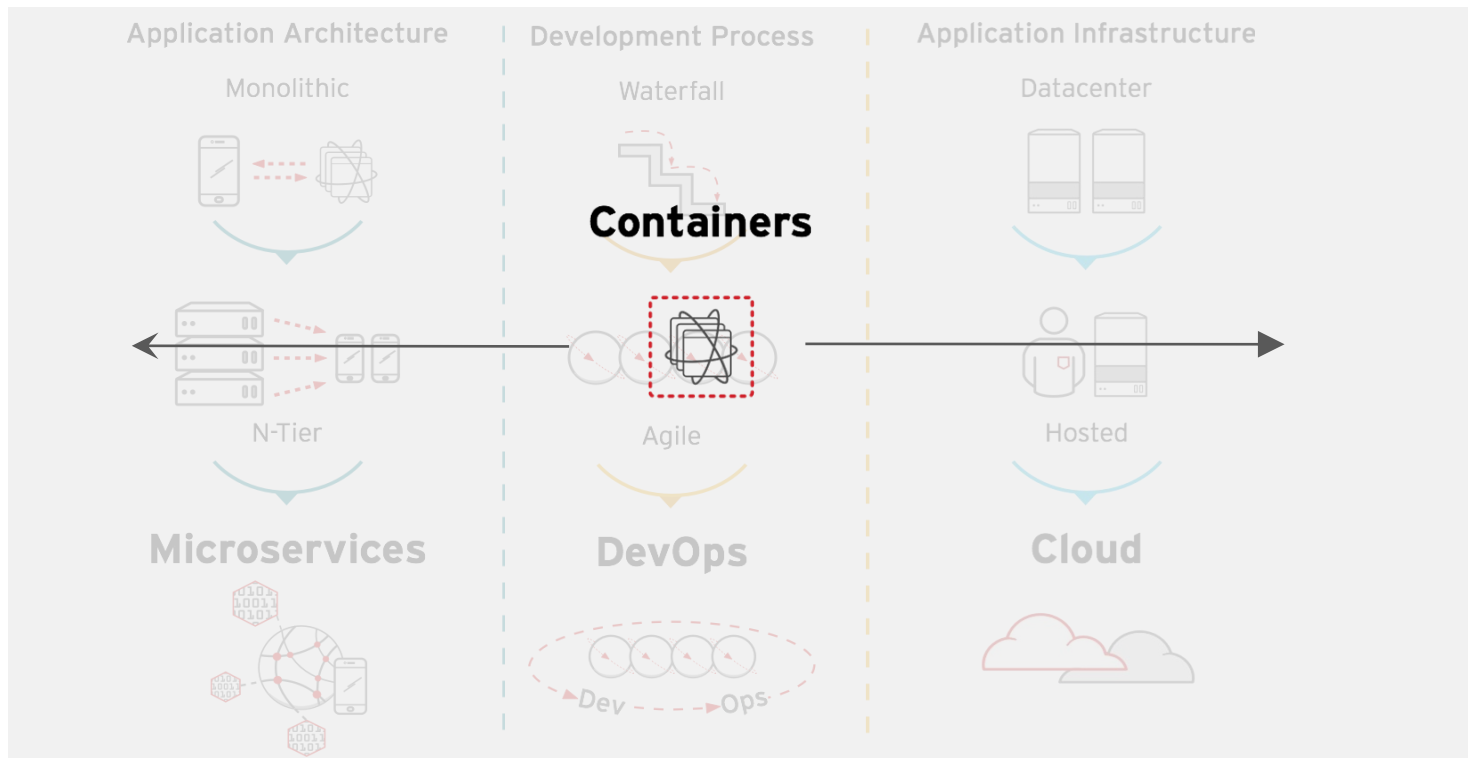


Hosted

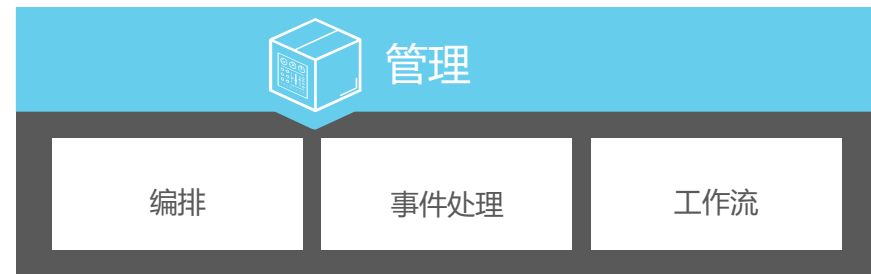
Cloud



容器-转型的基础



Devops 功能模型





- 规范化交付件
- 不可变基础设施
- 简化运维
- 部署和配置工作被提前到了编译时
- **Everything as code**





- 基础设施即代码
- **Source to image**
- 镜像管理和流转
- 环境版本控制
- 配置存放在环境中，动态更新





- 屏蔽基础设施差异
- 解耦应用和环境
- 开发，测试，生产环境的统一
- 支持不同开发语言和框架
- 存储、网络解耦
- 快速置备服务（数据库、消息等）





- 部署模板
- 服务关系管理
- 发布管理（灰度、蓝绿等）
- 容错
- 弹性
- 隔离





- 集中式日志、监控方案
- 健康检查
- 计费
- 报表
- 分布式跟踪



基于Docker和Kubernetes的容器管理平台，提供大规模集群上的容器应用的开发、部署和管理的功能。



- 调度
- 健康检查
- 配置管理
- CICD
- 日志监控
- 安全
- 应用服务
- 代码管理
- Source to Image
- 部署模型
- 弹性扩展
- 服务发现
- 镜像仓库
- 网络存储管理



基于容器的Devops实施步骤

- 基础架构：
 - 构建容器平台并将其与企业系统集成所需要的服务器和软件整合。
- 发布管理：
 - 实现应用部署和交付流程的标准化和自动化。
- 开发：
 - 包含开发人员构建容器化应用，将现有应用迁移至容器，或者开发使用容器编排的微服务所需要的流程和工具。



实用指导准则

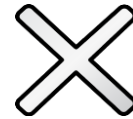
- 首先注重持续集成，然后构建管道
- 从小处着手：一个应用，一个团队
- 先实现容易达到的目标
- 确定指标和衡量标准
- 采用以成效为中心，而非以结果为中心的指标



更改的准备时间
更改的故障率
平均修复时间



代码行数
特性数量
代码覆盖范围



实用指导准则

- 快速迭代，并且尽快提供业务价值
- 以每次一步的方式扩展部署管道
- 展示可衡量的成功
- 向其他团队宣传，并且寻求志愿者进一步推广



主要结论

- 借助容器实现DevOps，可以改进质量和速度
- 在整个交付周期中使用容器带来了挑战，因为这不仅仅需要容器
- 选择合适的容器平台，从而为实现DevOps提供最快的方式



谢谢！

