应用DevOps的套路

刘征
DevOps教练

目录

DevOps is everywhere

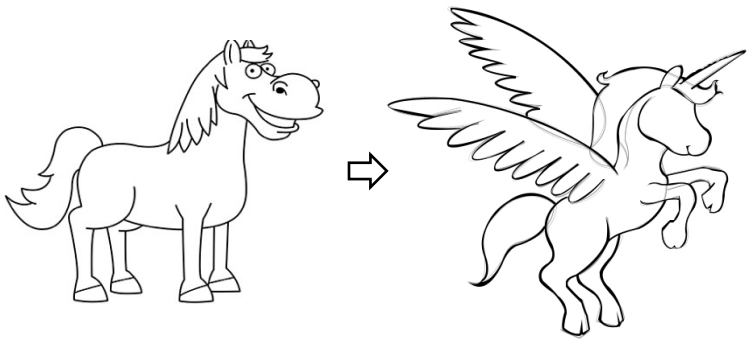
DevOps实施的常见反馈：

- DevOps也许在互联网企业推进更合适，传统企业实施阻碍太大
- DevOps的变革成本很高，需要采购一大批工具、调整组织结构
- 其他公司的DevOps实施路径是什么？我们就模仿他们的做法吧

DevOps转型的难点和误区

1 DevOps也许在互联网企业推进更合适，传统类型企业并不适合？

- DevOps实践被互联网公司所倡导
- 谷歌、亚马逊、Facebook、Etsy、LinkedIn …
- 百度、阿里、腾讯、京东 …



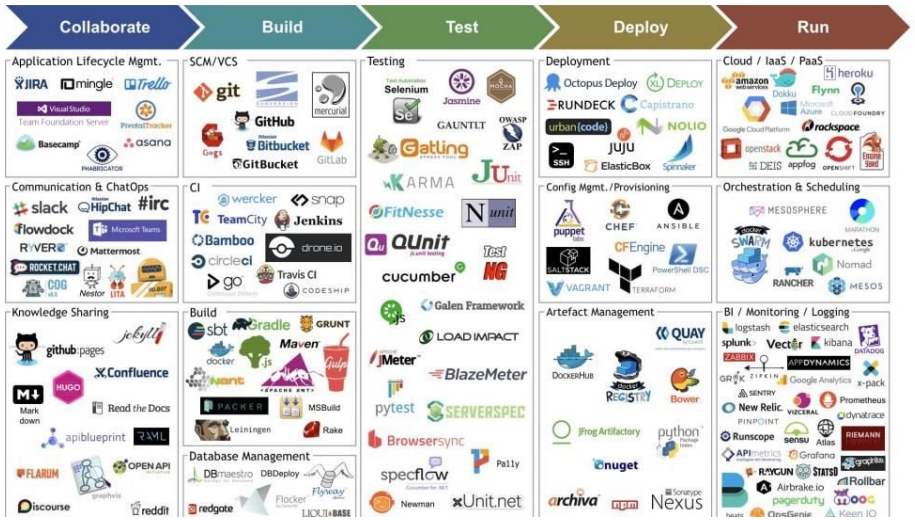
- 用发展的、演进的思维看待问题
- 正是因为这些公司重构了系统架构、技术实践，改变了文化的氛围，才能走向成功
- DevOps已被证明是普适的方法和技术
  - 银行、保险、电信、零售、甚至政府
  - Web、APP、嵌入式、甚至大型机

美国CapialOne银行转型数据	2016	2017
# Products deploying multiple times a day	~20	~300
Average #deployments per day	~1	~4
Max #deployments for a product in a single day	~30	~50

DevOps转型的难点和误区

2 DevOps的变革成本很高，需要采购一大批工具?

- 工具可以对DevOps的实施提供强有力支持
- 如果组织仅仅是购买工具而不改变工作流程，这样不会改变任何事情



- 以解决问题为核心，小批量持续改进
- 工具不是重点，重点是相关能力的建设

普遍问题一： 环境建立

- **问题**：测试或生产环境通常等待数周才建好
- **对策**：按需、完全自服务的创建环境

普遍问题二： 代码部署

- **问题**：代码部署通常通过多个手工、易出错的多人协作完成
- **对策**：自动化部署，目标是实现开发自服务，且完全自动化

普遍问题三： 测试执行

- **问题**：花两周准备环境和数据，四周进行手工回归测试
- **对策**：创建自动化测试集，让测试速度跟上代码部署

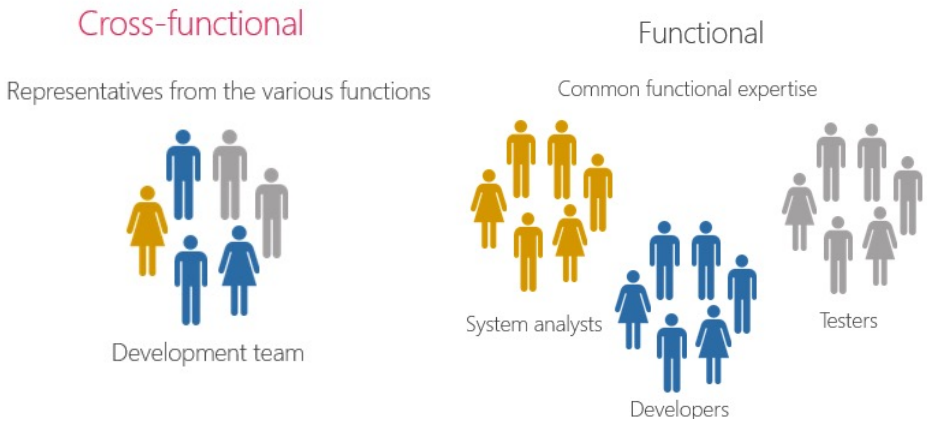
普遍问题四： 架构紧耦合

- **问题**：每次代码变更都要工程师到管理层或CAB进行申请
- **对策**：解耦架构，变更可以更安全和自治，提高开发生产率

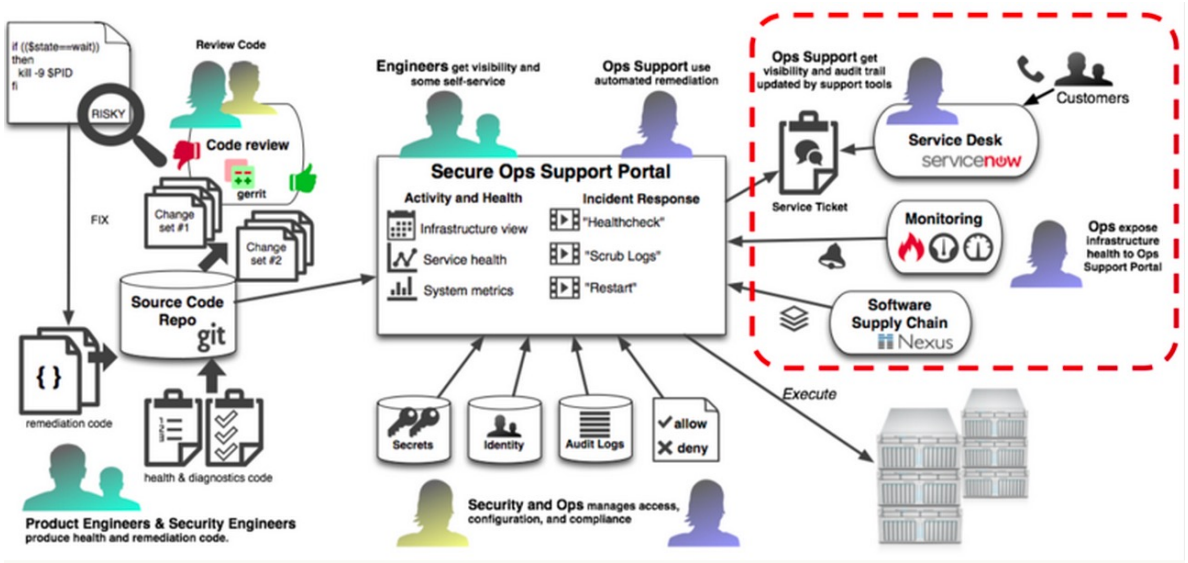
DevOps转型的难点和误区

3 DevOps需要大规模组织结构重组，否则无法实施？

- **职能导向的团队**：强调专业性和集中管理，等级式的组织结构，频繁工作移交、效率低
- **市场导向的团队**：强调快速响应用户需求，跨职能团队，可独立快速交付用户价值



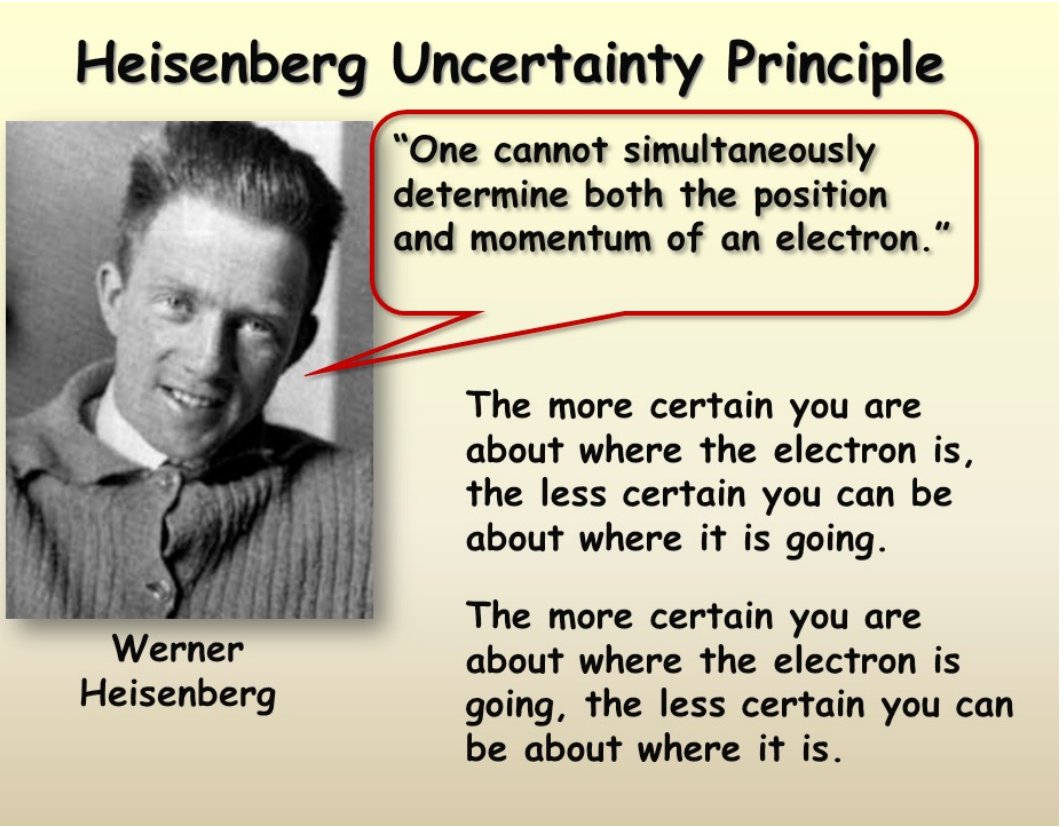
- 但组织结构重组（Re-Org） 并不是唯一选择
- 可以让不同角色物理地工作在一起，提示协作效率
- 下游职能团队通过自动化的自服务平台，给上游赋能



DevOps转型的难点和误区

4 工作系统是极其复杂的，工作环境是持续动态变化的该怎么办？

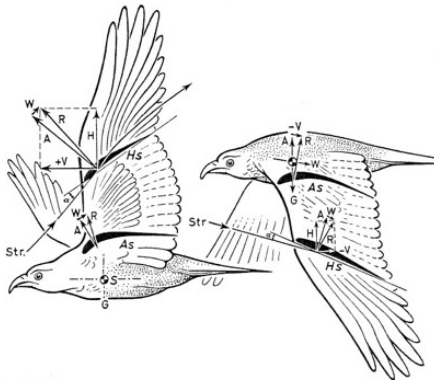
不可见的复杂系统

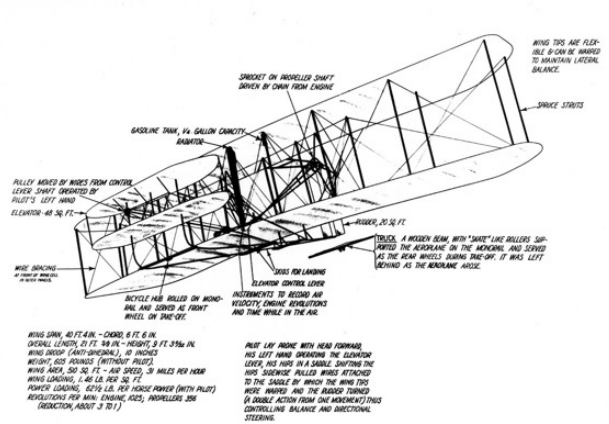


DevOps转型的难点和误区

5 其他公司的DevOps实施路径是什么？我们就模仿他们来做吧？

- 鸟飞派 vs 空气动力学派
- 初期的模仿是可以的，但很难照搬其他企业的做法而直接获得成功

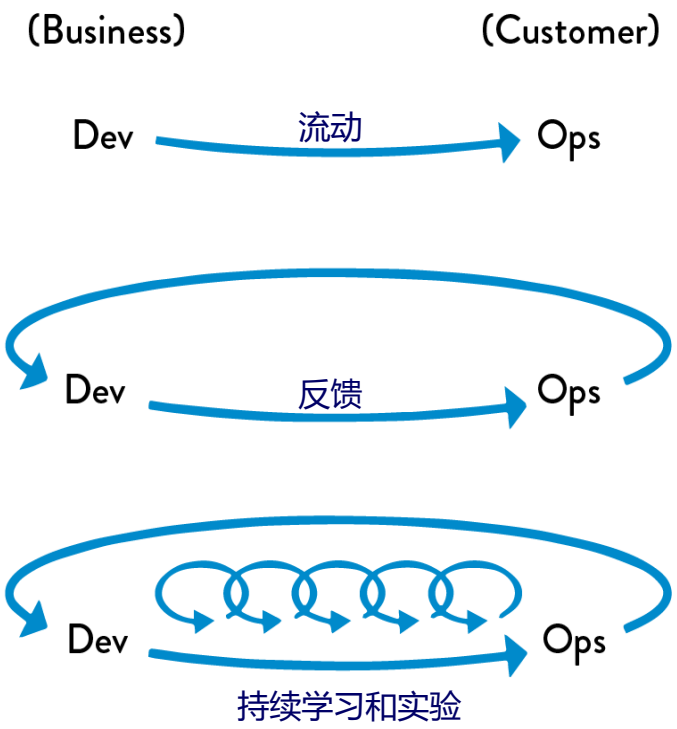




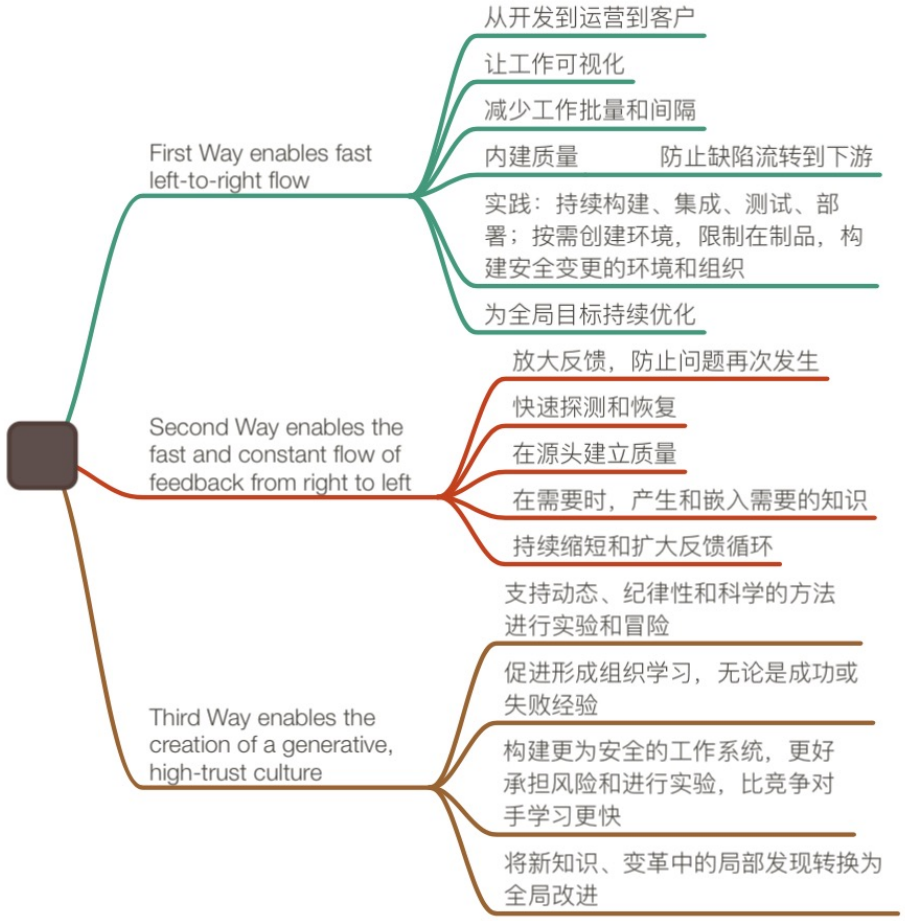
- 需要深入理解DevOps实施路径背后的原理、原则和实践，从正确的方向入手
- 可以从模仿开始，但最终要超越模仿

启动DevOps之旅的正确方法

遵循的心法和原则：DevOps的三步工作法

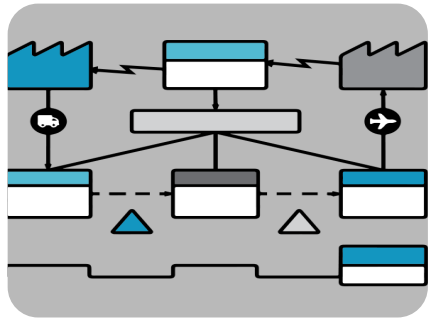


Source: DevOps Handbook, Chapter 1



启动DevOps之旅的正确方法

方法一：《DevOps实践指南》中的推荐的循序渐进的路径



选择价值流

- 绿地/棕地?
- SOR/SOE?
- 从创新项目开始
- 逐步扩大范围

可视化价值流

- 确定相关团队
- 绘制价值流
- 组建专门的团队
- 配套必要的工具

组建组织和架构

- 组建市场导向团队
- 招募通才团队成员
- 合理设计团队边界
- 保证松耦合和安全性

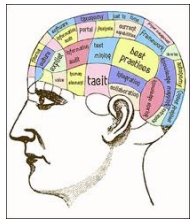
运维与开发融合

- 建立提高开发效率的自助服务
- 将运维融入开发日常工作
- 在团队中配置运维人员

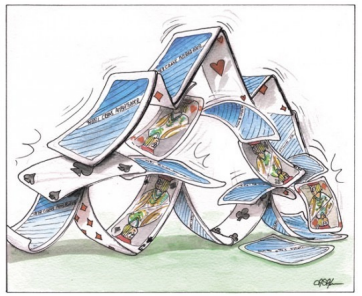
启动DevOps之旅的正确方法

方法二：《精益思想》中推荐的行动计划

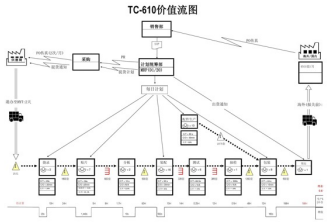
1. 寻找变革代理人



2. 获取精益知识



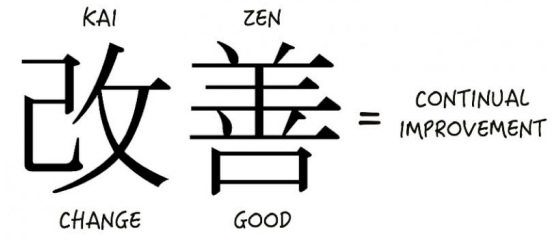
3. 寻找危急-险中求胜



4. 绘制产品的价值流图

5. 根据价值流图重组公司

6. 创建精益促进机构



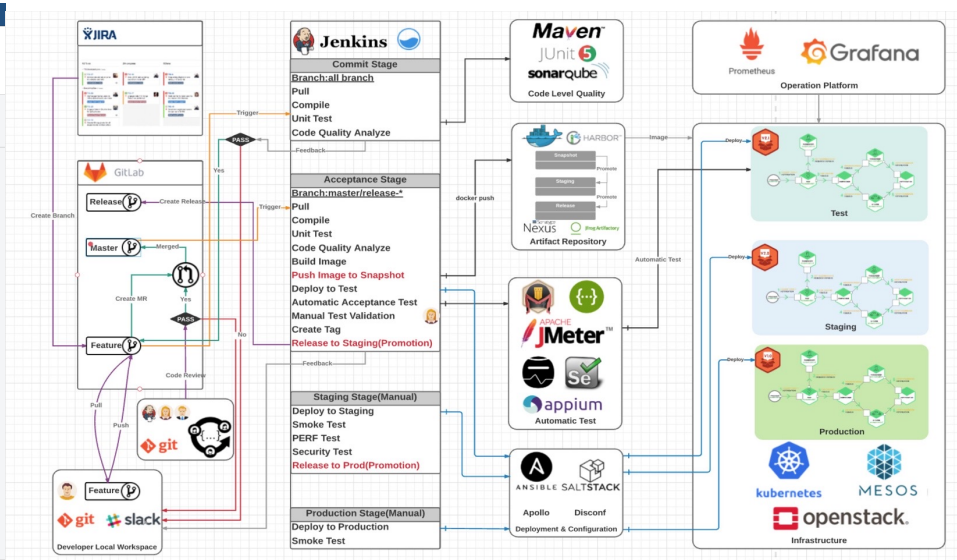
7.

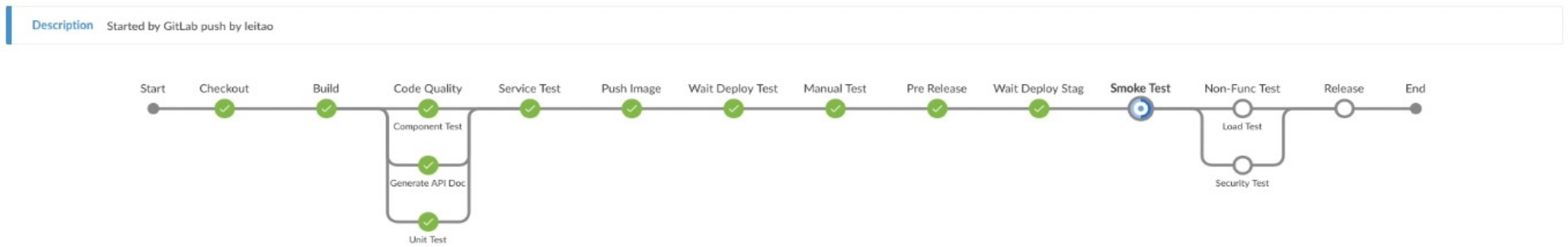
8. 利用政策为实施保驾护航

9. 同化你的供应商和客户

启动DevOps之旅的正确方法

案例：管理维度 + 工程维度的同步推进





案例：美国全国保险公司的DevOps登山指南

- 所有开发团队实施前置时间
- 授权团队自行选择路径
- 朝每日、零宕机部署努力

登山装备

New Relic
urban[code]
GitHub
Jenkins
splunk>
sonarqube
ROCKET.CHAT
Hand Book

- Handbook是地图
- 用地图规划路径
- 工具乃登山必备
- 工具皆各自有用

BASE CAMP 大本营

- 从运用敏捷开发开始
- 应用各种DevOps工具
- 成立DevOps Handbook读书俱乐部
- 秉持持续改进程序员体验的观念

NORTH CAMP 北坡营地

- 实施代码和数据的自动化部署
- 实施暗发布模式
- 在代码就绪/校验之后正式对外发布
- 推送流水线状态/健康度至Rocket.Chat
- 通过聊天机器人最小化上下文切换
- 发展多元化技能
- 实施去中心化的性能测试、安全测试和部署
- 实现生产环境的自动化校验和反馈
- 通过价值流映射的方式定位约束点
- 使用前置时间指标驱动持续改进

SUMMIT 顶峰

- 实现每日发布的频率
- 实施ChatOps
- 业务伙伴创建MMP并根据反馈指标而调整
- 实现单件流

SHERPAs 夏尔巴人

- 技术顾问
- DevOps教练

COMING SOON

- 沉浸式配对
- 每月DevOps道场
- 交付流水线支持

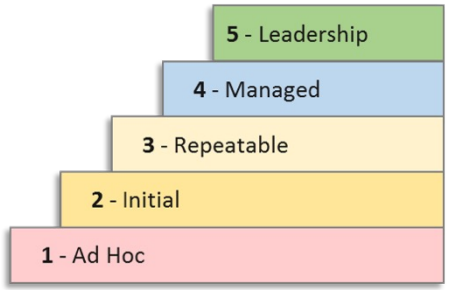
支持模型

DevOps的实施能力和效果如何评估?

DevOps的实施能力和效果应该如何评估？有没有通用的成熟度模型?

成熟度模型的特征:

- 阶梯式，统一的成熟度，所谓的标准?
- 定义技术、过程和组织能力的静态等级
- 组织的目标是“通过”某个级别



成熟度模型的局限:

- 企业环境、约束不同，级别定义都是相同的?
- IT行业每年都在发生着剧变，但模型不变化?
- 仅度量技术和工具本身，如何体现对结果影响?
- 达到了某级就会失去之前的资源和预算?
- 锚定和满足感阻挡了探索和尝试?
- **还有没有更好的方法?**

Maturity models are for CHUMPS

- 2005年，角色级别上限升至70，重新设置角色职业，增加新专业技能、副本、战场和种族，可以变更角色
- 2008年，角色级别上限升至80，增加死亡骑士职业，开放更多副本和新的装备
- 2010年，角色级别上限升至85，新增两个种族，游戏的内容在改头换面的艾泽拉斯大陆进行，增加新专业技能

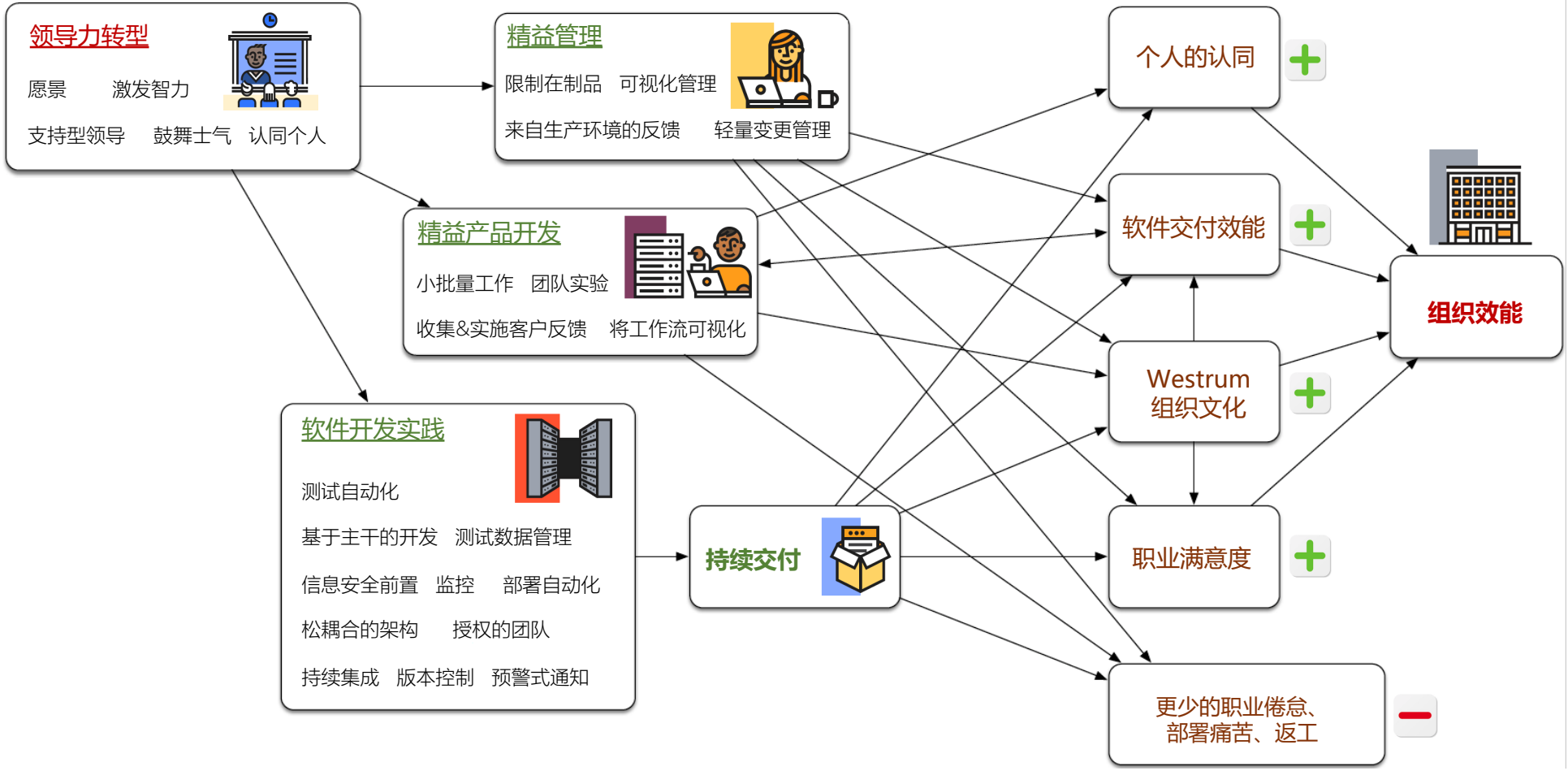
Maturity models are for CHUMPS

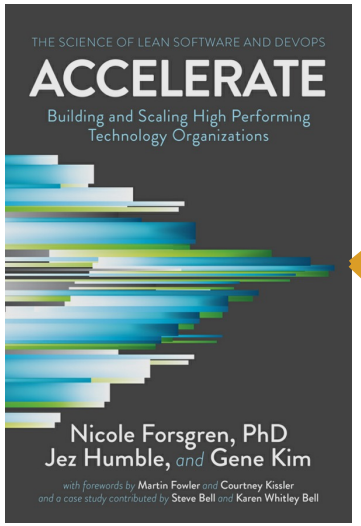
- 2011年，角色级别上限升至90
- 2013年，角色级别上限升至100，新增新的大陆德拉诺
- 2017年，角色级别上限升至120，部落和联盟接在海上找到了新的盟友

业务和技术环境每年都在变化

DevOps的能力成长模型

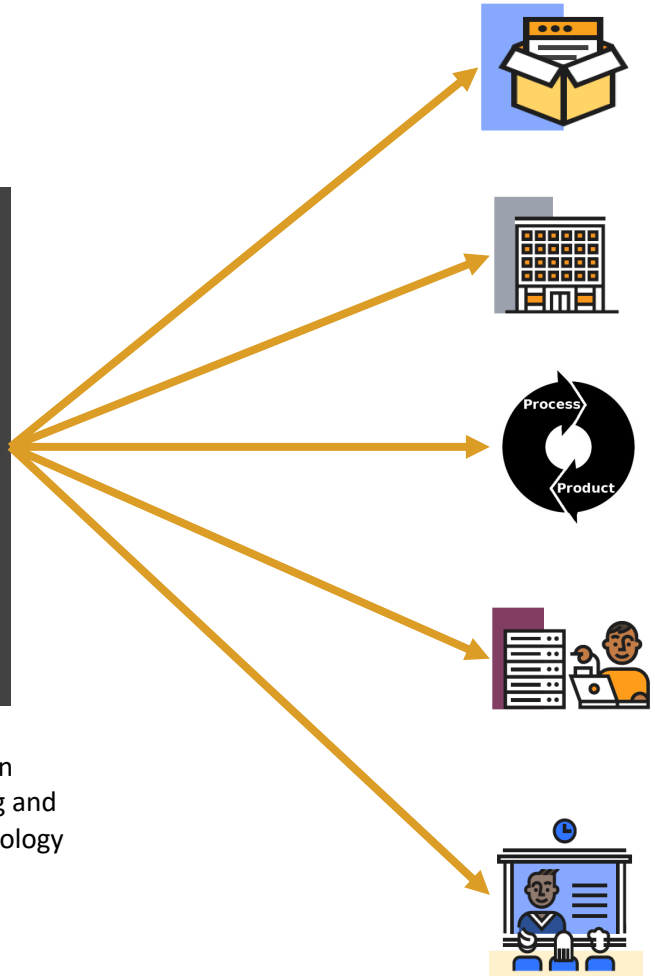
- 适用于本来能力和起点就参差不齐的不同团队
- 符合每个团队的优劣势和业务上下文的不同
- 每个团队聚焦在不同的、各自的约束点/弱点上
- 根据行业基线设置下一步的目标





Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations

Link: http://a.co/eo5FyQj



分类		DevOps能力描述
持续交付	1	对生产环境的所有工件进行版本控制
	2	运用自动化部署流程
	3	实施持续集成
	4	运用基于主干的开发方法
	5	实施测试自动化
	6	进行测试数据管理
	7	前置信息安全管理
	8	实施持续交付
系统架构	9	进行松耦合的架构设计
	10	授权团队作出架构设计决策
产品和流程	11	收集和实施客户反馈
	12	用价值流将工作流可视化
	13	工作在小规模的批量中
	14	培养和赋能团队进行试验
精益管理和监控	15	应用轻量的变更审批流程
	16	用从应用到基础架构支持业务决定
	17	有前瞻性地检查系统监控状况
	18	通过WIP来概述流程和管理工作
	19	通过可视化来监控和沟通团队的工作质量
企业文化	20	发展并支持生机型文化
	21	鼓励和支持学习
	22	支持和辅助团队间的协作
	23	提供工作所需要的资源和工具
	24	支持和落实领导力转型

如何持续学习和持续改进

在复杂系统中实施DevOps仍然是很有挑战的，如何能够更快的进步?

- 实现飞行的关键技术是升力、动力和控制
- 莱特兄弟进行上千次的风洞实验，不急于试飞
- 思维的飞跃：摆脱模仿，从原理出发进行发明

- DevOps实施过程的实验和测试非常重要
- 需要一整套持续改进的机制或方法
- 建立实验的组织文化
  - Google的灾难恢复测试
  - Etsy的"三只袖毛衣"奖励
  - Netflix的Chaos Monkey和Simian Army

TOYOTA Kata 的持续改进之道

Learner
学习者

1

Get the Direction or Challenge

3

Establish your Next Target Condition

Obstacles

Threshold of Knowledge

2

Grasp the Current Condition

4

Experiments

Coach
教练

Better Coaches
Better Players
Better Game

COACHING KATA

The Five Questions

1) What is the **Target Condition?**
2) What is the **Actual Condition** now?

--------*(Turn Card Over)*---------->

3) What **Obstacles** do you think are preventing you from reaching the target condition?
   Which *one* are you addressing now?
4) What is your **Next Step**? (next PDCA / experiment) What do you expect?
5) When can we go and see what we **Have Learned** from taking that step?

*You'll often work on the same obstacle for several PDCA cycles