# Gdevops

# 全球敏捷运维峰会

## 开发运维一体化PaaS平台建设

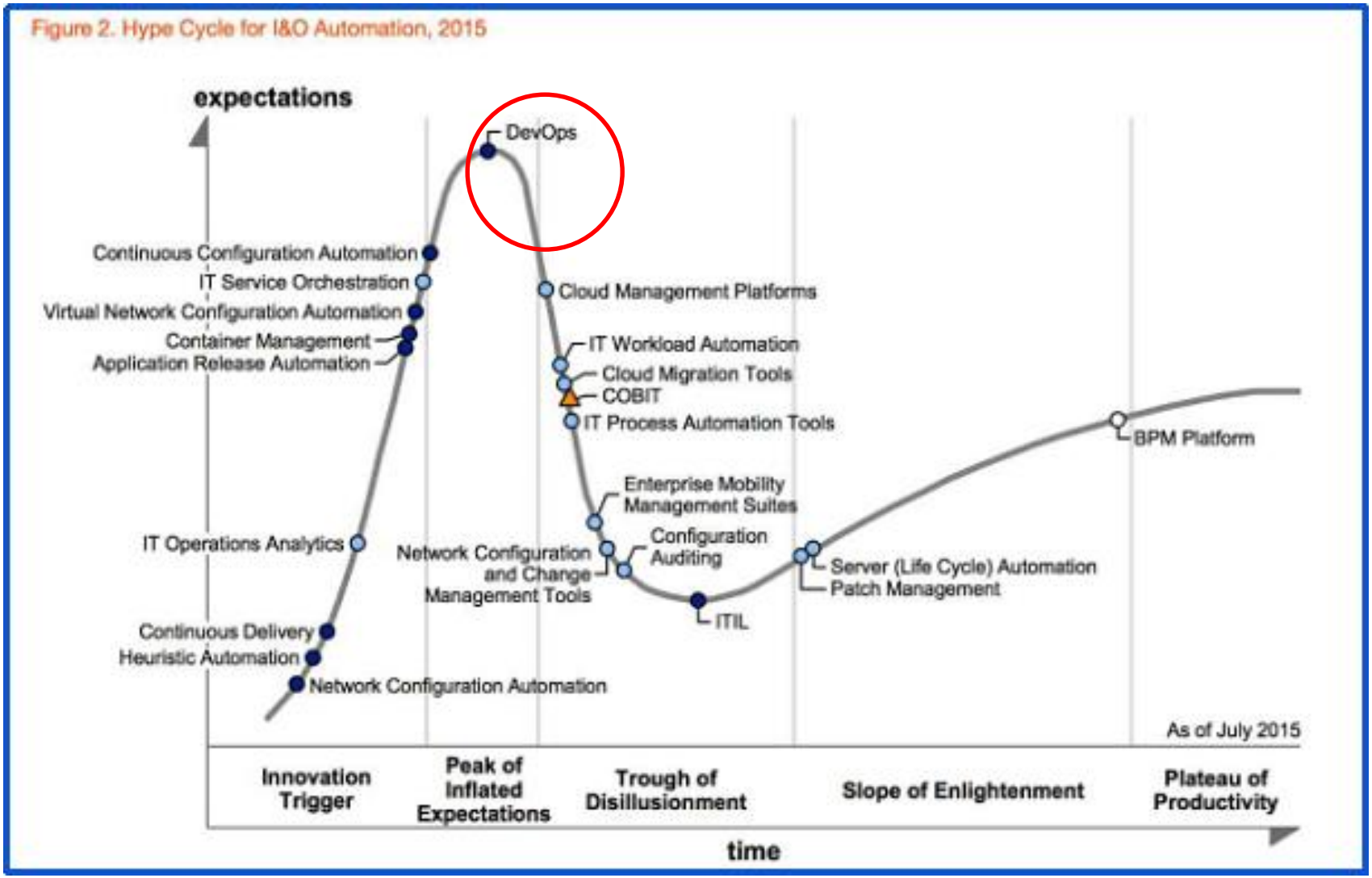
演讲人：陈能技

在过去一年中大约推送了**5000**万个部署，每分钟达到**95**至**100**个。

平均部署时间：11.6秒

部署失败停机率：0.001%

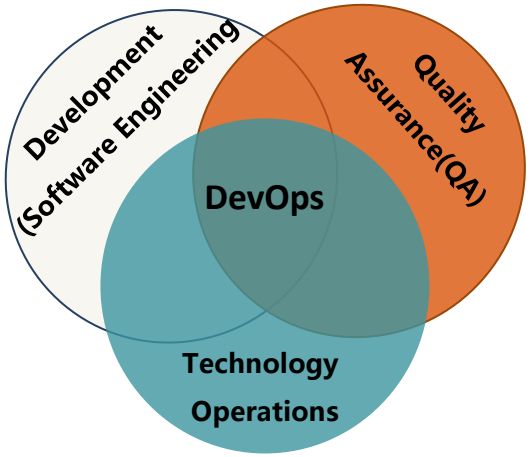
# 根据Gartner2015年的调查报告，DevOps正处在技术发展曲线的最高点

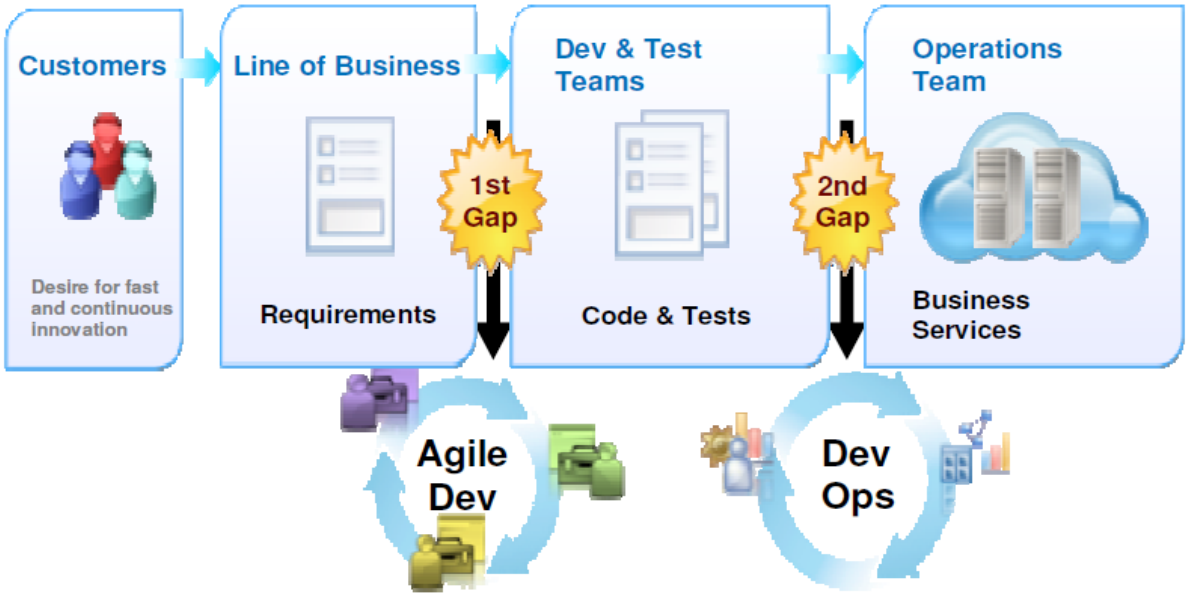


# 敏捷开发打破第一个Gap，DevOps帮我们打破第二个Gap！

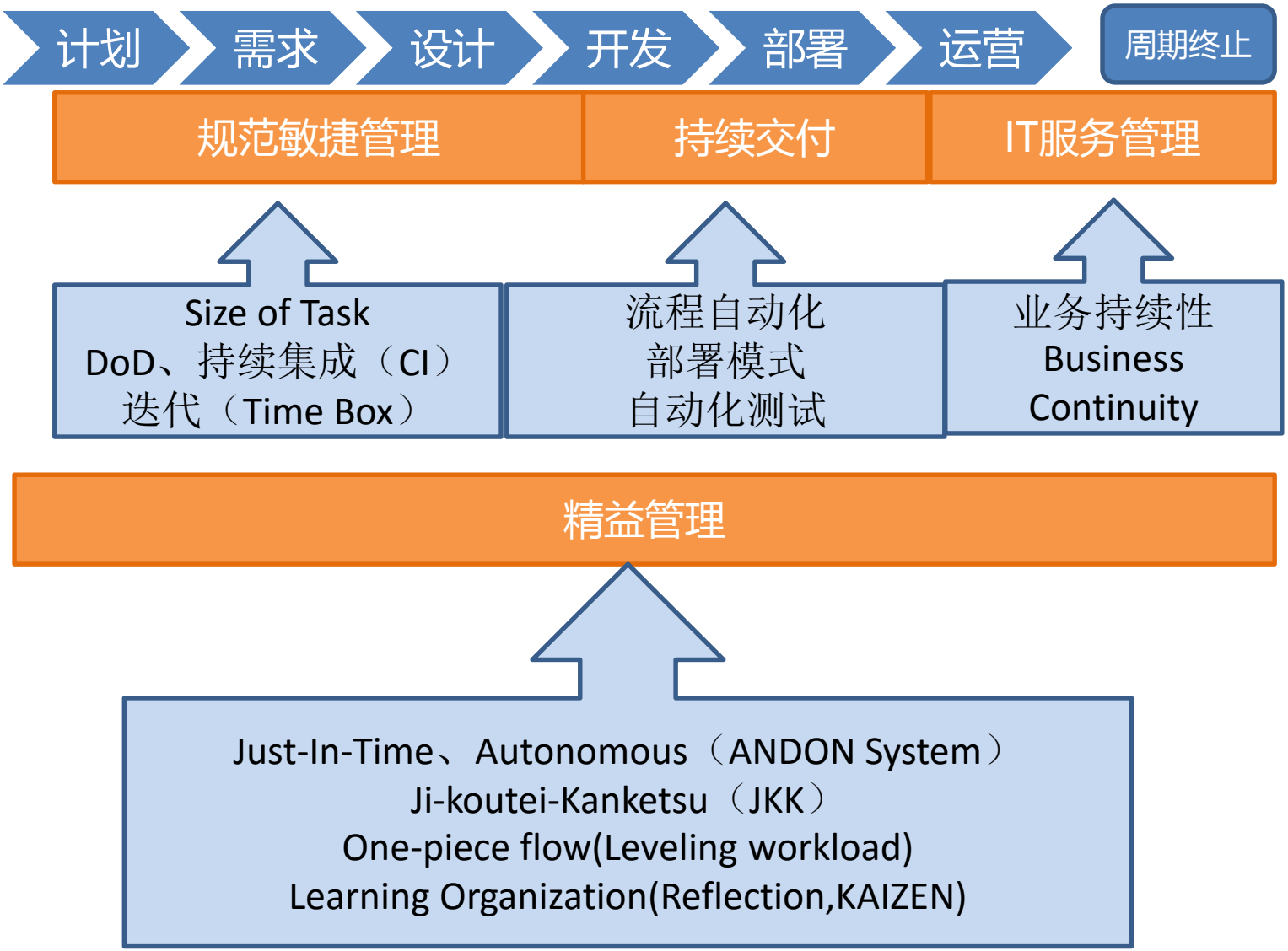
研究显示，在那些引入了DevOps概念的企业中，开发与运营人员在设计、构建、测试工作中共同在内部应用上进行协作之后，交付能力大大提升：

交付效率提高**30**倍、变更失败率降低**50%**

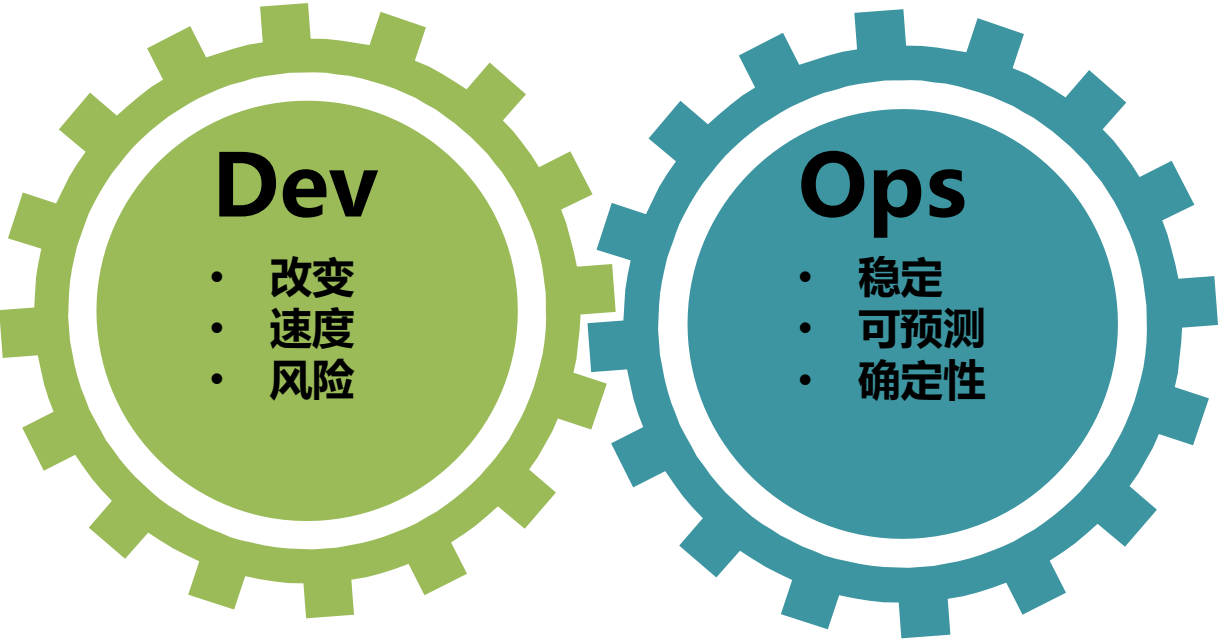


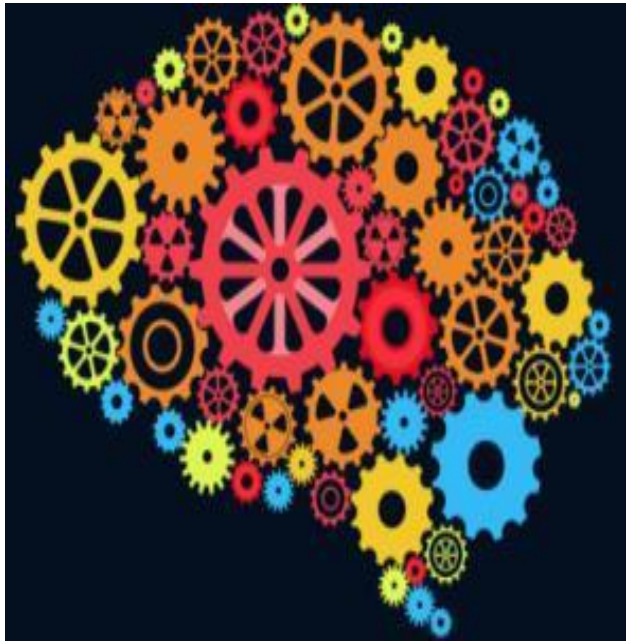



# DevOps尝试整合已有的IT能力



# DevOps旨在通过共同目标，成就协同与效率，提升用户体验



# 新炬提出DevOps能力矩阵模型

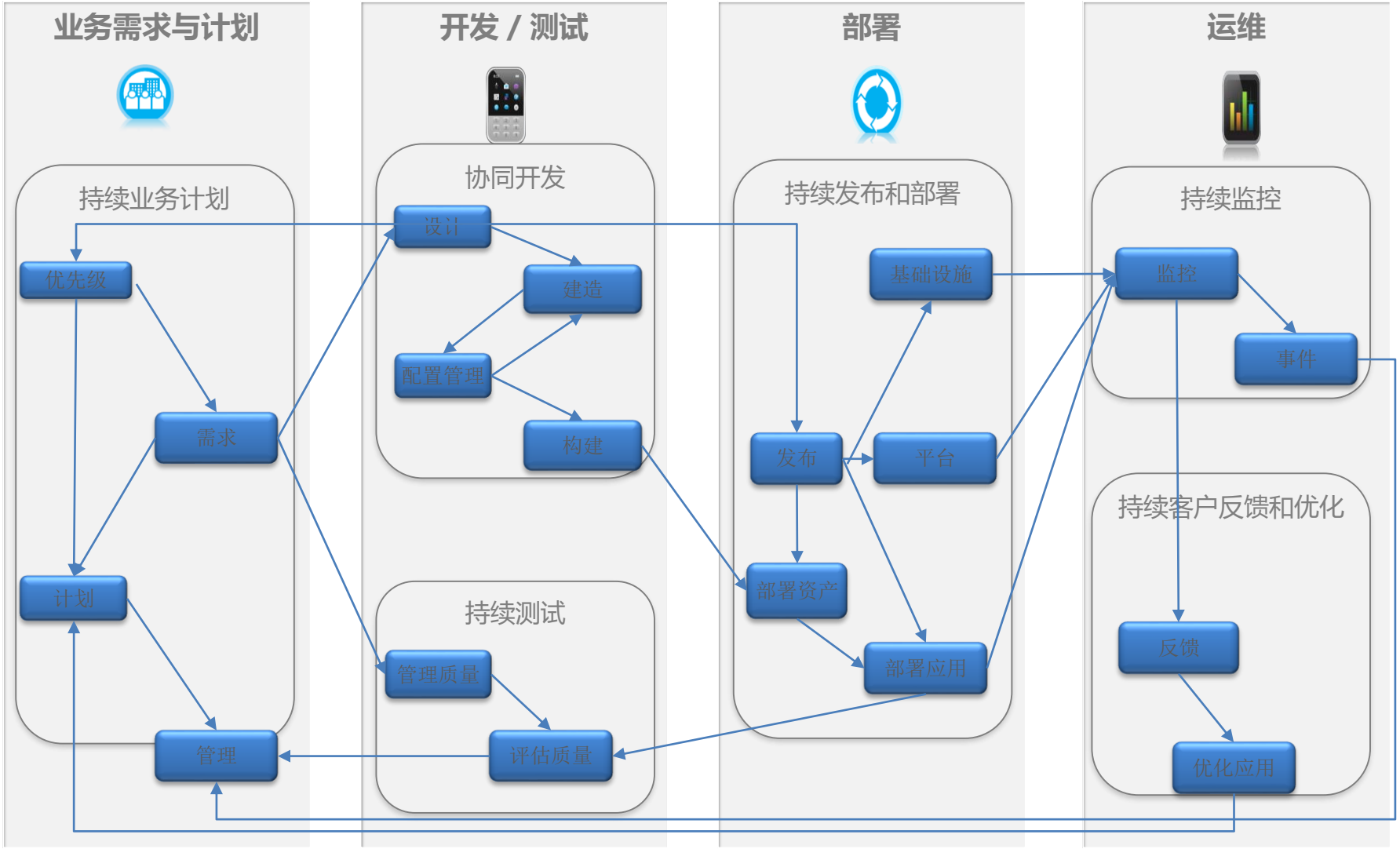
## 角色视角 X 管理视角

传统企业的DevOps转型关键在于角色能力的转型。

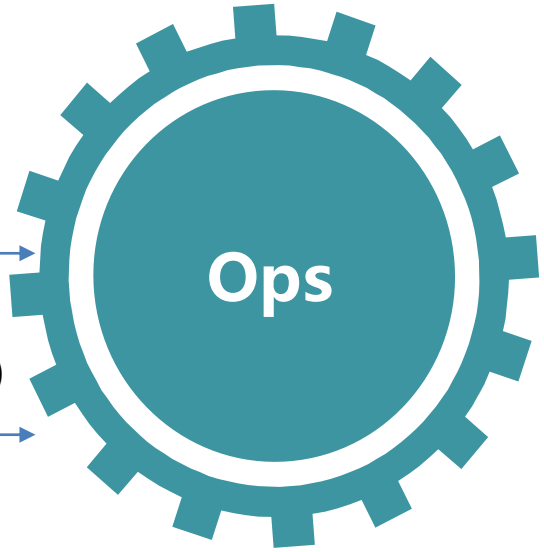
| | 度量 | 过程 | 工 具 |
|---|---|---|---|
| 开发 | 代码质量<br>技术债务 | 小版本迭代开发<br>持续集成<br>持续部署 | Docker微服务<br>自动部署 |
| QA | 系统测试<br>测试覆盖率度量<br>验收测试 | 持续测试<br>测试环境管理<br>质量反馈 | 自动化测试<br>应用性能管理 |
| 运维 | 运营质量<br>运行质量<br>用户体验<br>趋势分析 | 持续发布<br>性能管理<br>容量管理<br>运维管理 | 一体化监控<br>ITOA大数据分析<br>自动化运维<br>应用性能管理 |

# DevOps能力融合

通过各类角色的能力融合、能力传递，减少流程环节的浪费，提高效率。



# DevOps能力融合4大核心实践



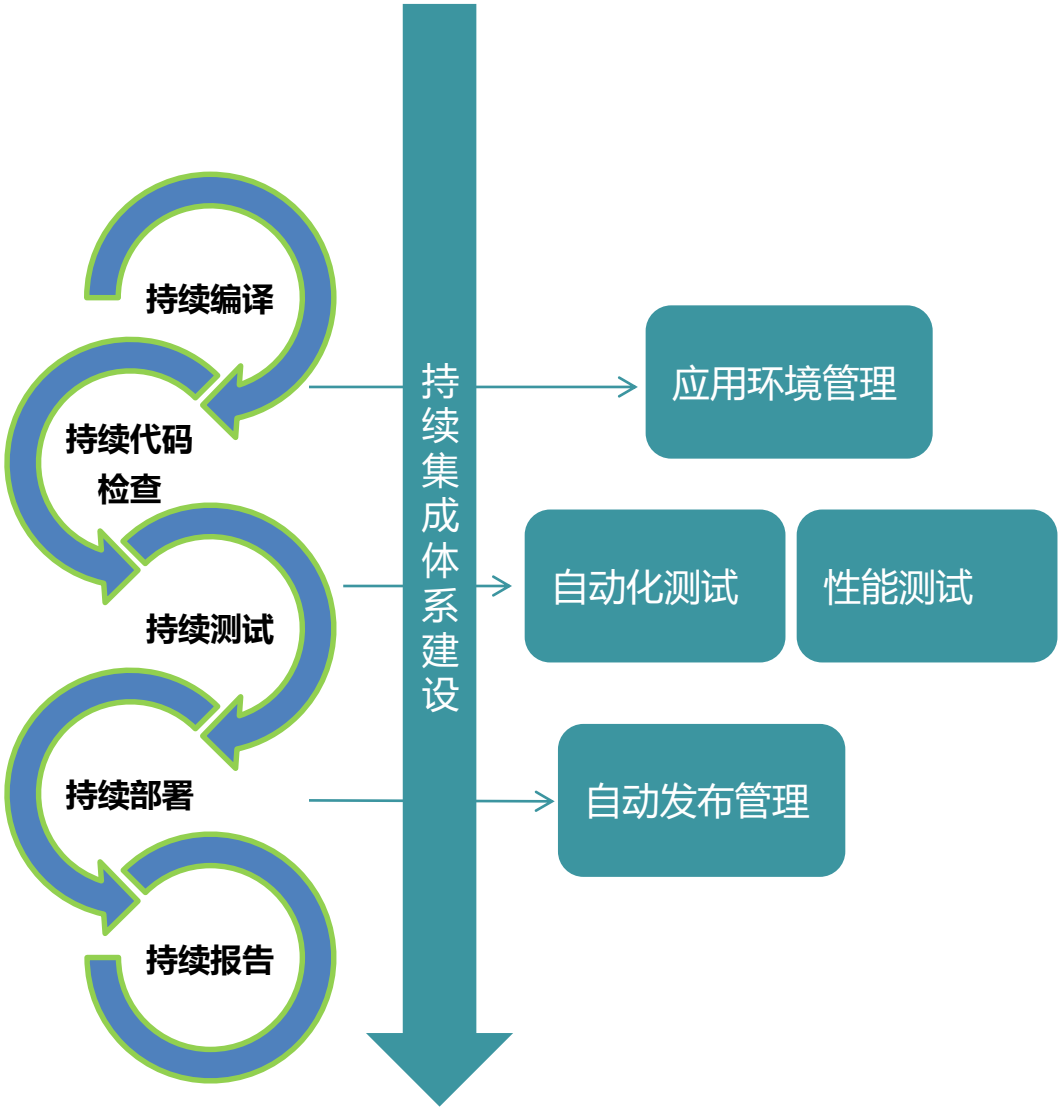

实践1：将开发延伸至生产中（持续集成和交付)

实践2：将开发嵌入到IT运维中（应用端到端管理）

实践3：向开发中加入生产反馈（可视化监控和运维）

实践4：将IT运维嵌入至开发（运维分析及预测）

# 实践1：建立持续集成体系，实现交付过程标准化与透明化




## 分阶段实施

1 制定基本构建流程

2 加强代码自动验证环节

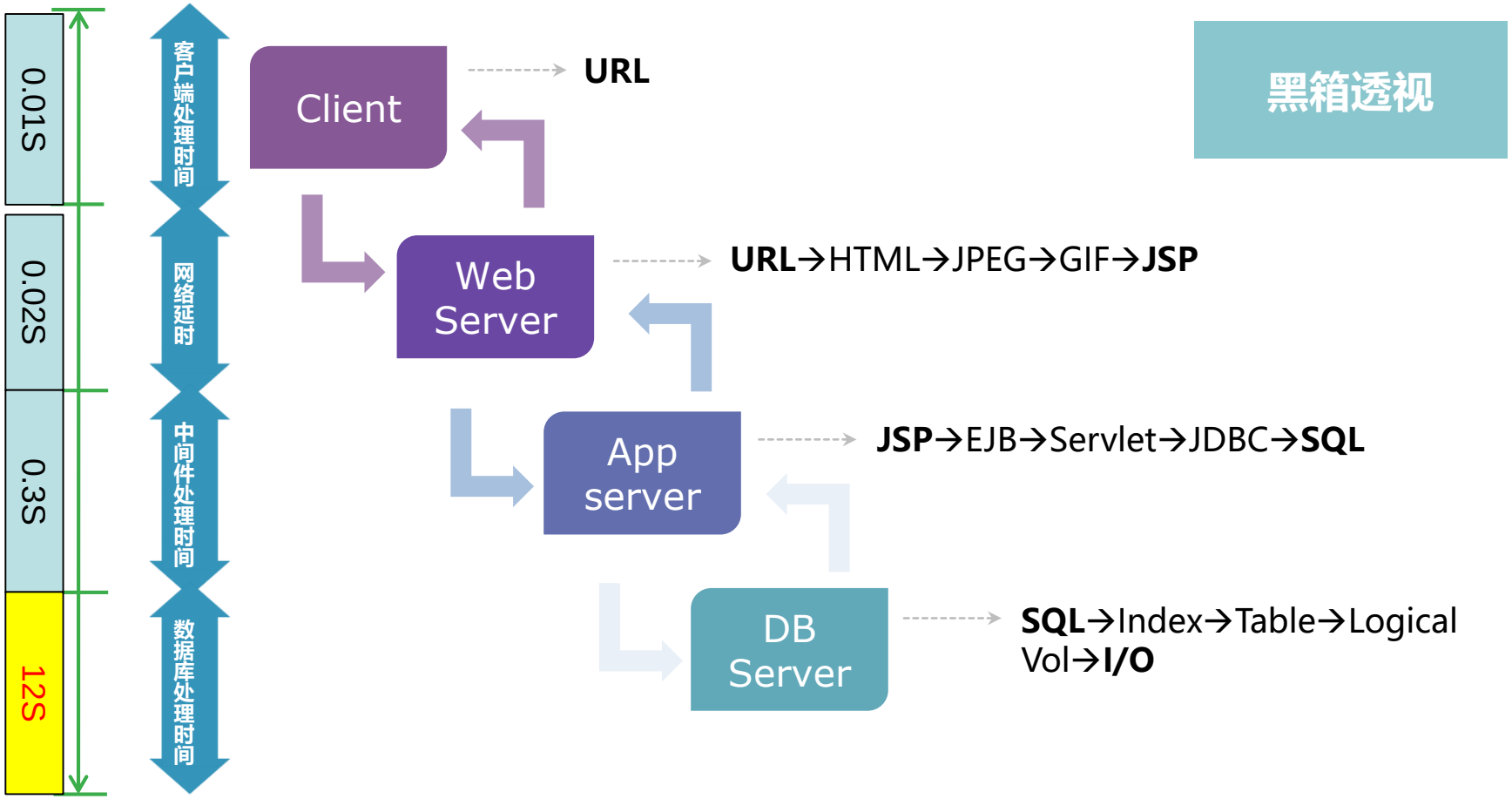
3 加强单元测试环节

4 加强自动化测试环节

5 加强自动部署环节

# 实践2：透明化应用交易过程，实现端到端的应用性能管理

通过对交易过程的追踪和记录，实现交易过程的透视，并对相关数据进行分析和存储，实现调用过程分解及性能问题的快速定位。



# 实践3：构建立体化监控体系，实现运行状态可视化及深度性能分析

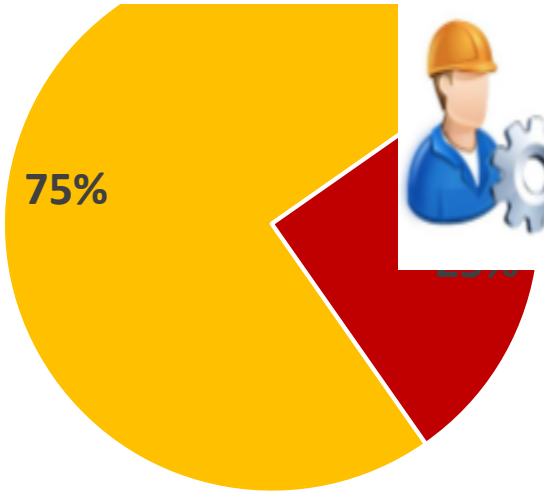
**向开发加入生产反馈**：开发关注应用层故障，运维需要快速排查应用层以下的故障，给开发精准反馈应用层问题，降低开发排故的诊断分析难度，节省排故时间。

- 用户
- 业务
- 应用
- 中间件
- 数据库
- 主机
- 存储/网络

# 实践4：整合利用运维数据，进行运维大数据分析

7年全世界将有75%的大企业会积极
进行如，可用性监控，应用性能监控，
,容量分析等

开发人员：分析和诊断应用程序缺陷，优化代码质量，提高终端用户的体验质量。



运维人员：快速定位故障根源，降低故障恢复时间，提升SLA。

、应用程序、传感器产生的日志

运营人员：通过对业务日志进行分析及挖掘，有效实现产品及企业的运营，实现商业价值。

最大力度挖掘数据价值

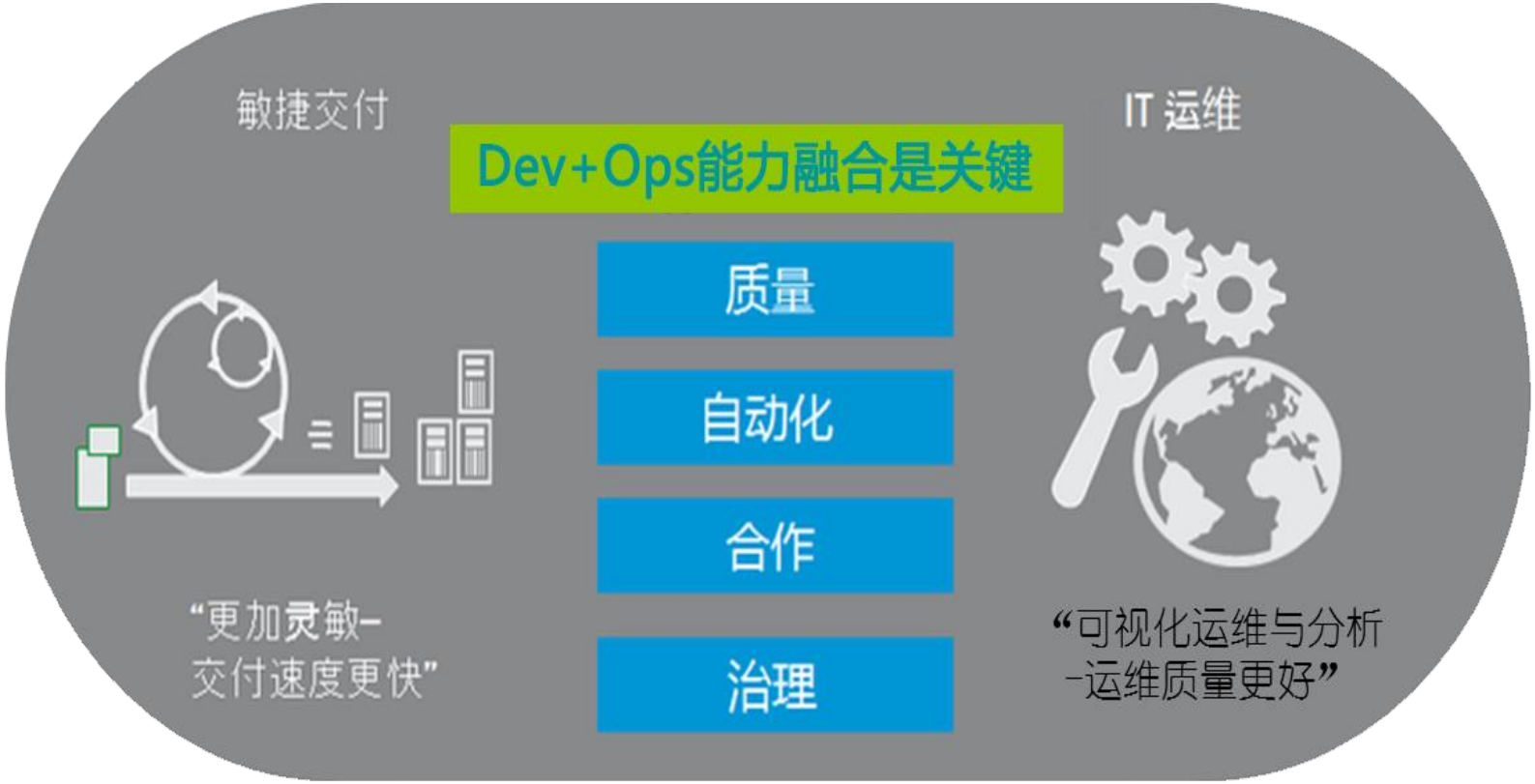
D 监控数据（M
-- 监控采集到的

E 探测数据（P
-- 模拟用户请求

安全人员：通过对海量安全日志的分析，过滤出安全事件，查找安全隐患，保障系统安全运行。

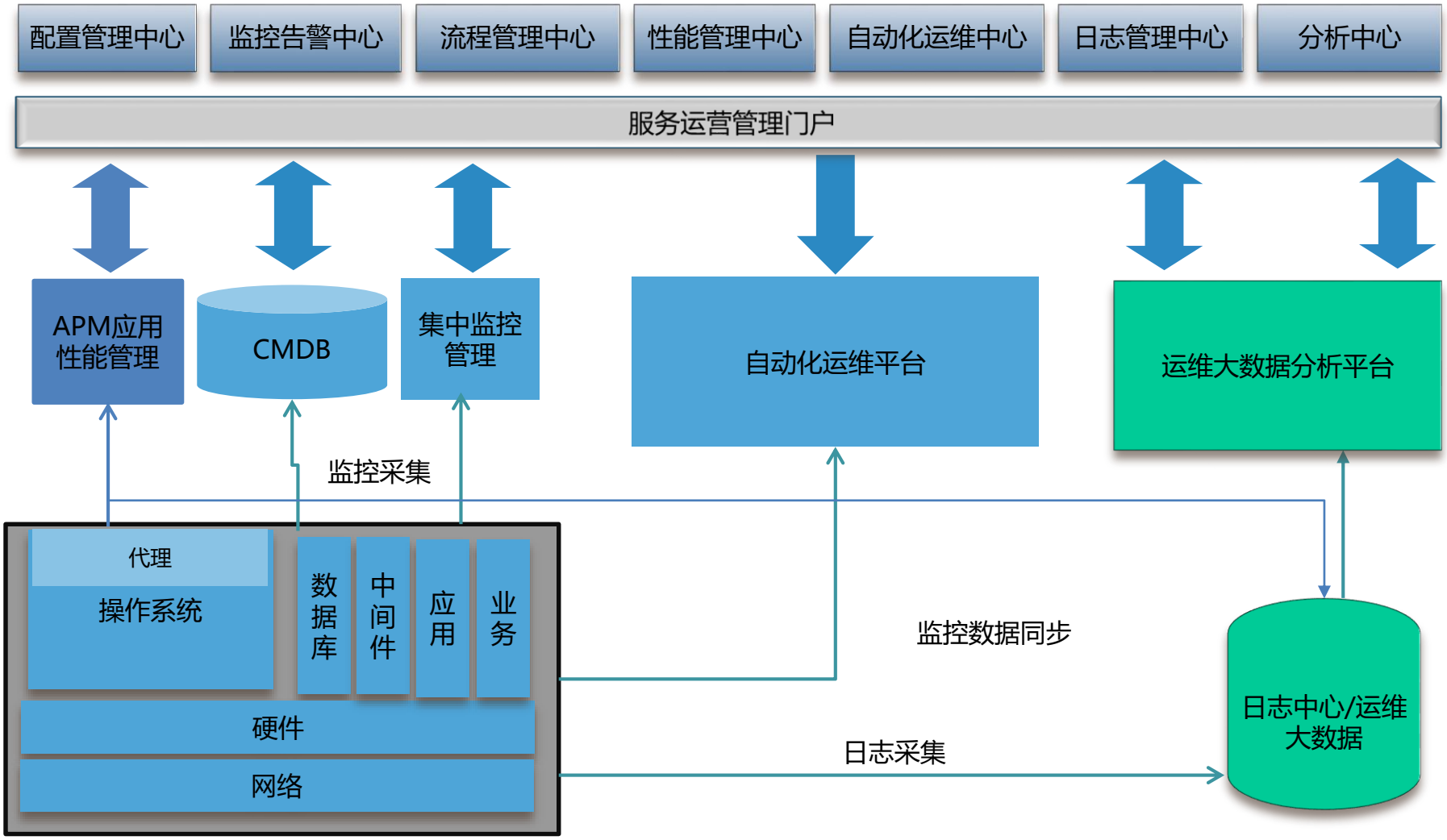
# 基于DevOps能力矩阵模型，实现从敏捷交付到敏捷运维

角色的能力融合是关键。
开发到交付运维的管理过程交叉融合是基础。
基于新兴技术（例如CI、CD、Docker、APM、日志分析等）构建开发运维一体化PaaS平台是趋势。



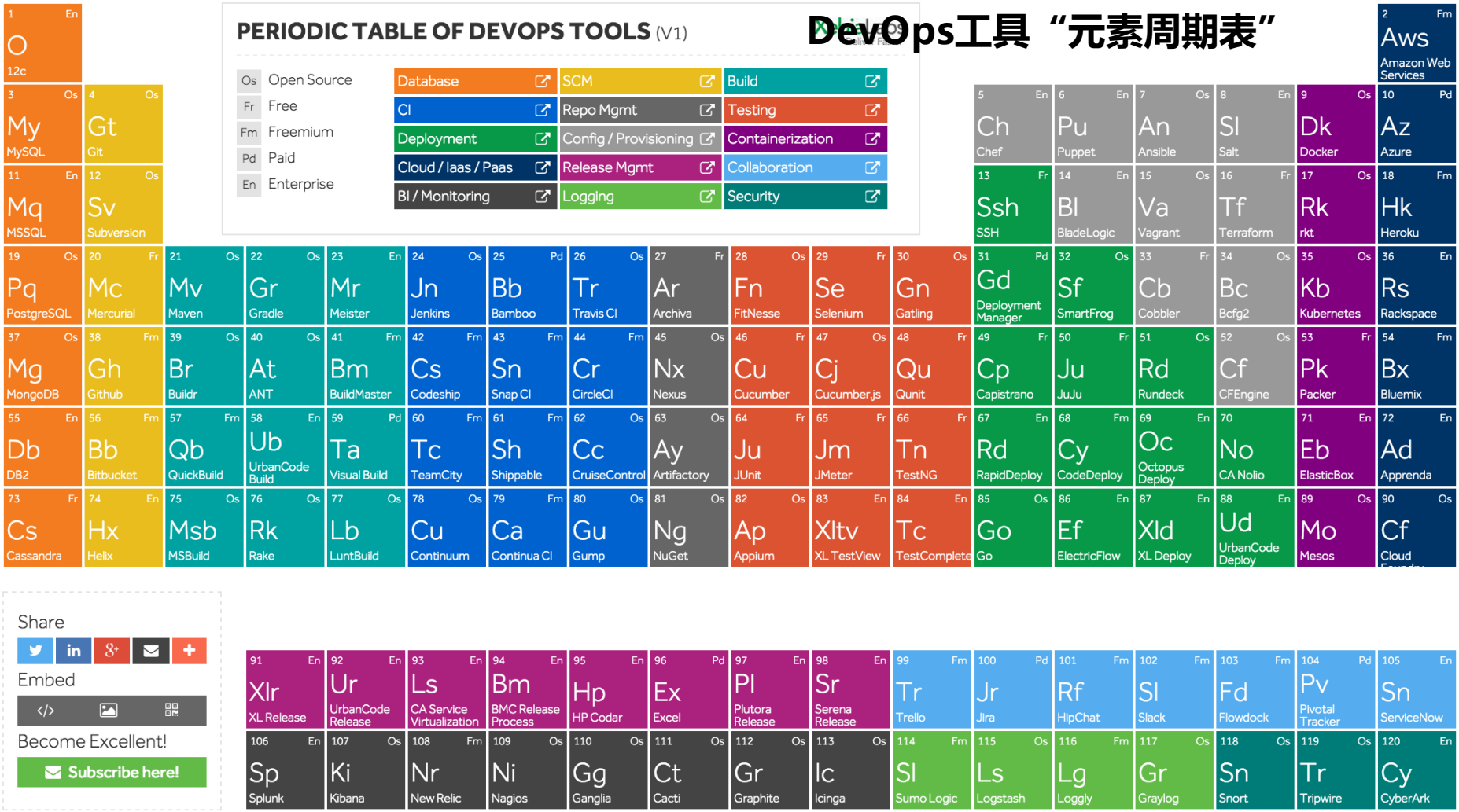
# 构建DevOps全过程工具链，打造传统企业的“蓝鲸”体系

## DevOps：可视化、数据化、自动化、智能化

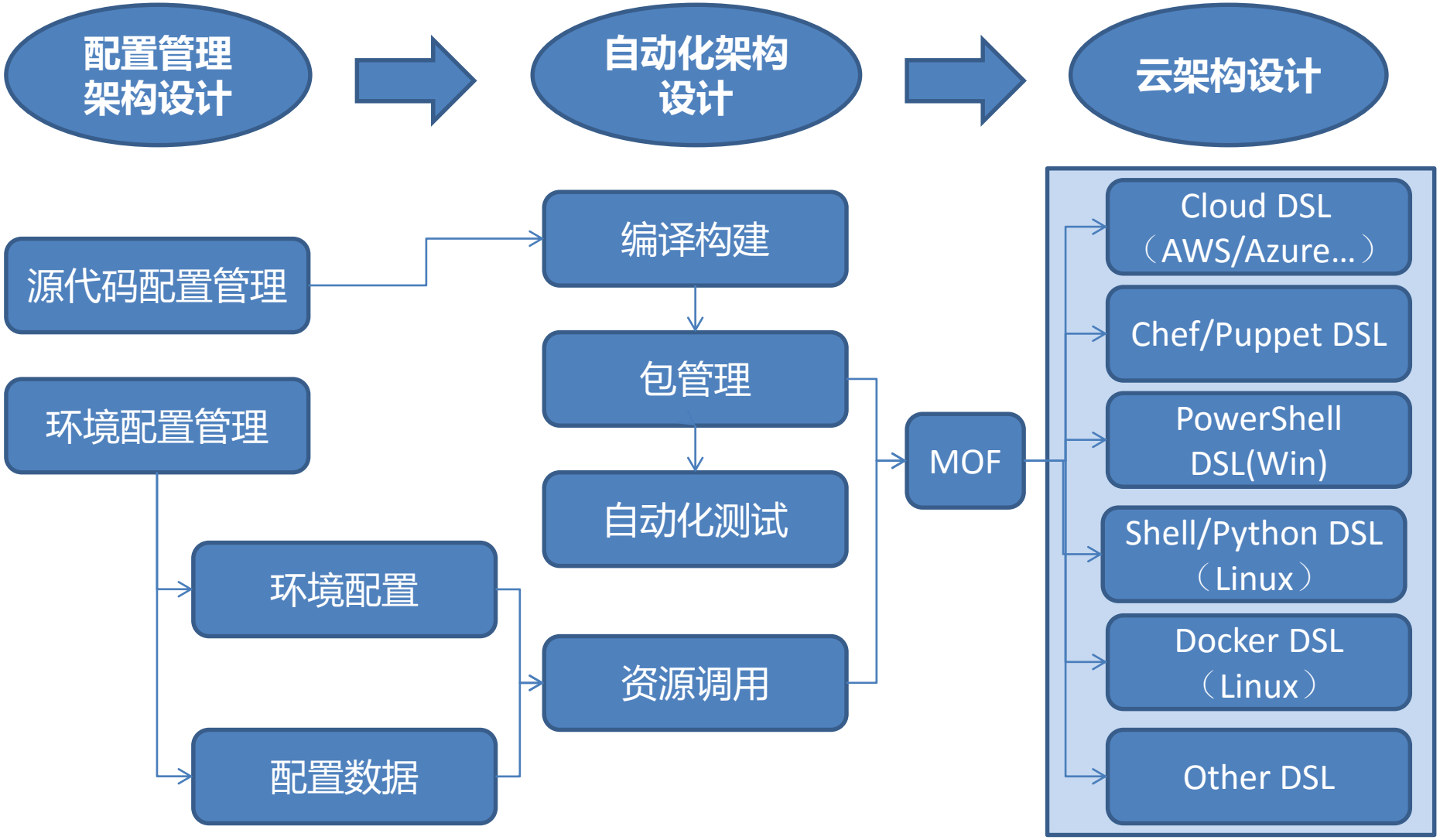


# 开发运维一体化PaaS平台建设：工具整合能力

流程的自动化是关键，能通过整合工具完成的事情就别用人做。



# 开发运维一体化PaaS平台建设：三大核心基础架构设计



# 开发运维一体化PaaS平台建设：流程规范化能力

基于DevOps成熟度模型制定流程规范，借助工具平台落实。

| 实践 | 构建管理和持续集成 | 环境和部署 | 发布管理和合规 | 测试 | 数据管理 | 配置管理 |
|---|---|---|---|---|---|---|
| **3级-可优化级：**聚焦于流程改进 | 团队经常碰面讨论集成问题和自动化解决方案，更快速的反馈和更高的透明度 | 所有环境都被高效地管理起来，自动化地配置管理，虚拟化应用程度高 | 运维和发布团队协作管理风险和减少周转时间 | 几乎没有生产回退，缺陷能及时发现和修复 | 数据库性能和部署过程在两次发布之间构成反馈循环 | 经常性地检验CM策略能支持高效的协作快速开发和可审计的变更管理流程 |
| **2级-可量化级：**过程度量和控制 | 收集构建度量信息、数据可视化并采取行动，确保构建不被破坏 | 各类部署过程和编排都被管理起来，发布和回退过程被充分测试 | 主动管理环境和应用监控，周转时间被收集记录起来 | 质量度量和趋势跟踪，非功能需求被定义和度量 | 每次部署都验证数据库更新和回退，数据库性能被监控和优化 | 开发人员每天至少签入一次代码到主线，分支只被用于发布 |
| **1级-可持续级：**过程自动化应用到整个应用生命周期中 | 每次变更提交都触发自化构建和测试，依赖包得以管理，重用脚本和工具 | 全自动化，提供自助服务式部署都程 | | 自动化单元和验收测试，后者由开发过程伴试活动 | 作为部署过程的一个环节，数据库变更自动化执行 | 库和依赖被管理起来版本控制的使用策略由变更管理流程决定 |
| **0级-可重复级：**流程文档化，部分自动化完成 | 自动化构建和测试，任何一次构建都可以使用自动化流程从源代码控制开始重新创建 | 自动化部署到某些环境，创建一套新环境比较简单快捷，所有配置都版本化、外在化 | 布过程比较痛苦，但发布过程本身还是比较可靠的，从需求到发布仅有限的可追溯性 | 部分自动化测试 | 数据库变更由版本化的自动化脚本执行完成 | 所有重建软件系统所需的东西都纳入了版本控制：源代码、配置、构建和部署脚本数据迁移 |
| **-1级-退化级：**过程不可重复，缺乏控制 | 纯手工过程构建软件，工件缺乏管理和报告 | 纯手工过程部署软件系统，二进制文件跟环境相关，环境配置手工进行 | 不能频繁发布，发布过程不可靠 | 手工测试 | 数据迁移手工进行且未版本化 | 未使用版本控制，或签入代码不频繁 |

大部分企业处于可重复级以下，需要往更高级别发展！

# 开发运维一体化PaaS平台建设：支持DevOps团队组织架构

康威定律：产品必然是其组织沟通结构的缩影。

**流程主管（Process Master）**
Scrum Master，敏捷项目领导（Agile Project Leader）。

**服务主管（Service Master）**
Scrum产品负责人（Scrum Product Owner）、服务负责人（Service Owner）。

**DevOps工程师（DevOps Engineer）**
开发（Development），工具（Tools）。

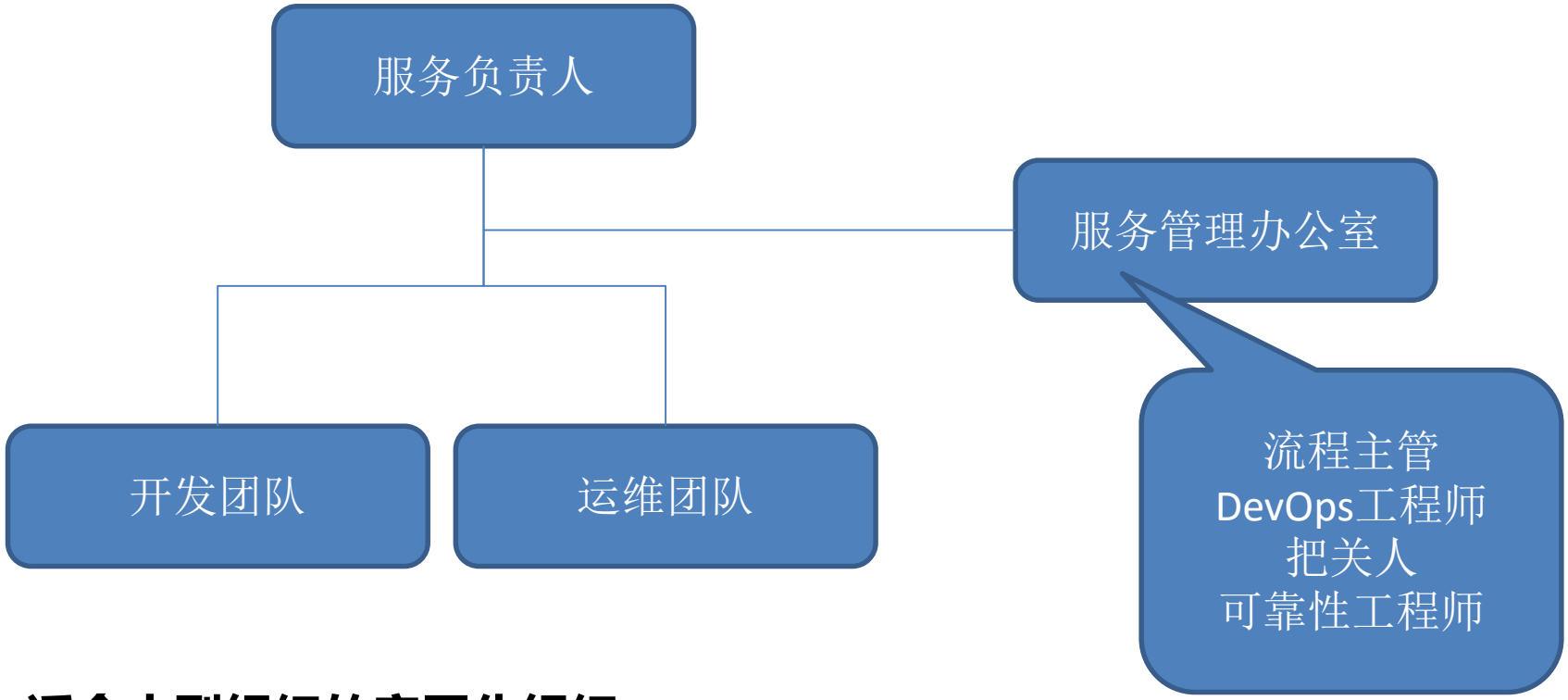
**把关人 / 发布协调员（Gatekeeper / Release coordinator）**
IT服务管理（IT service management）、运维（Operations）。

**可靠性工程师（Reliability Engineer）（可选）**
测试（Testing），工具（Tools），质量保证（Quality assurance）。

**开发团队（Development team）**
开发（Development），敏捷（Agile）

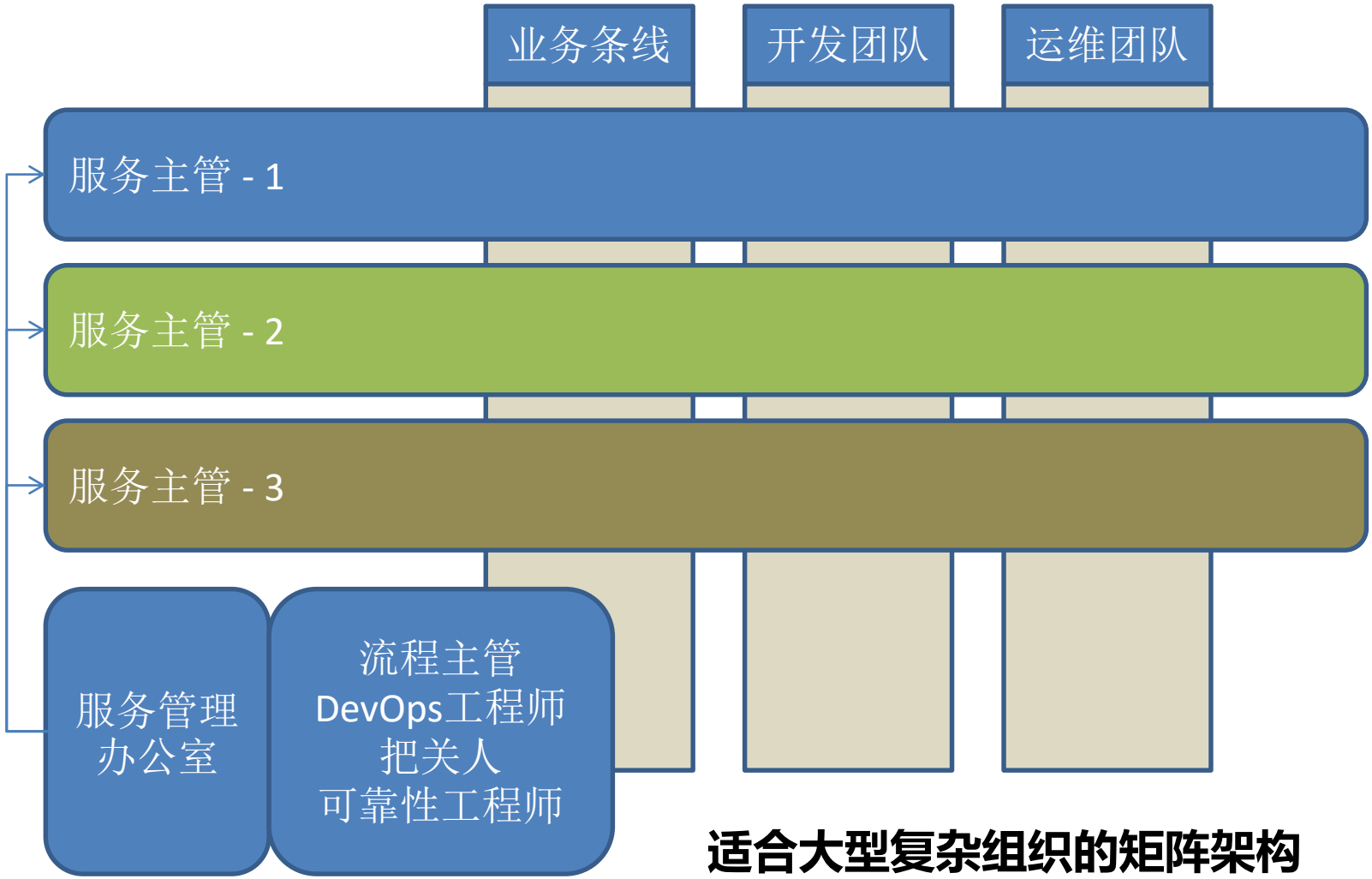
**运维团队（Operation team）**
运维（Operations），持续改善（KAIZEN）

# 开发运维一体化PaaS平台建设：支持DevOps团队组织架构




**适合小型组织的扁平化组织**

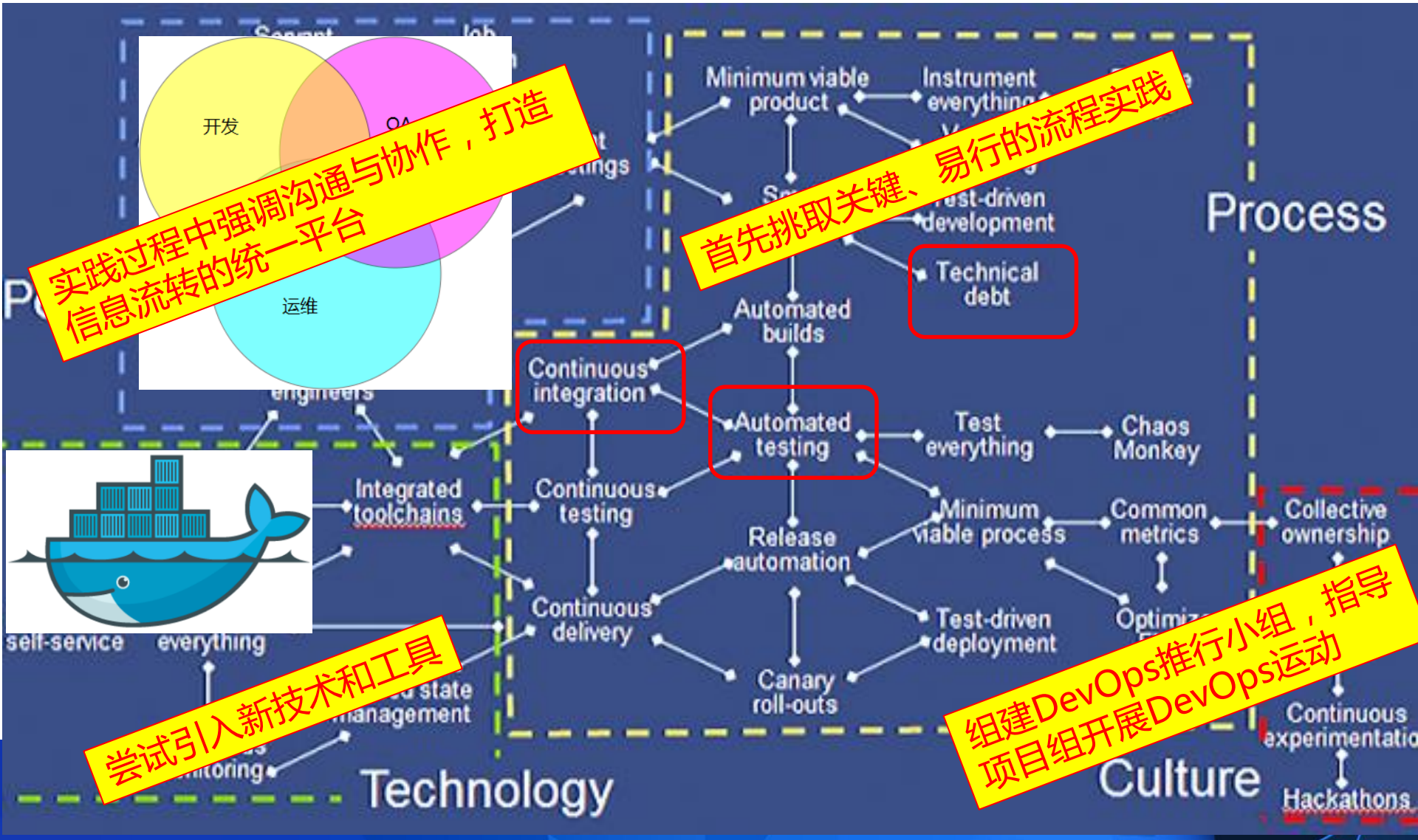
# 开发运维一体化PaaS平台建设：支持DevOps团队组织架构




**适合大型复杂组织的矩阵架构**

# 开发运维一体化PaaS平台建设：支持DevOps循序渐进的应用模式

DevOps应用实践四要素：人、流程、技术、文化，每一部分都需要建设的过程。



# Gdevops

# 全球敏捷运维峰会

THANK YOU！